UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

LENGUAJE DE PROGRAMACIÓN C APLICADO A EXPERIMENTOS ELÉCTRICOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

OLIVARES ESPINOSA ULISES

ASESOR: ING. FRANCISCO RAÚL ORTIZ GONZÁLEZ

MÉXICO **2008**

AGRADECIMIENTOS

Le doy gracias a mi mejor amigo que es Dios por todas las bendiciones que me brinda, y aunque ande en valle de sombra no temeré mal alguno, porque el está con migo.

A mis hijas PENÉLOPE OLIVARES MIRANDA y SARAHÍ OLIVARES MIRANDA, que son el encargo más hermoso e importante en mi vida, que Dios me ha dado.

A mi esposa AZUCENA MIRANDA SANDOVAL, a la que amo y comparto mi vida, y de quien recibo su ternura y apoyo.

A mis padres, señor HÉCTOR OLIVARES GUERRA y señora MARÍA ESTELA ESPINOSA REYES, que me llevaron de la mano en el camino de la vida y de mis estudios, guiando mis actos por el sendero del bien.

A mi hermano IVÁN OLIVARES ESPINOSA, que juntos hemos compartido momentos inolvidables y me ha apoyado en momentos importantes.

Con especial respeto y admiración para mi abuelo FLORENTINO OLIVARES RAMÍREZ† de quien obtuve la asesoría y motivación necesaria para el estudio.

A mi abuelita CASIMIRA GUERRA NEGRETE, a quien quiero y admiro por sus sabios consejos.

A mi maestra y tía MARGARITA OLIVARES GUERRA, con la que inicie mi preparación hacia mi futuro, guardando gratitud y afecto.

Con respeto a mi amigo, ING. FEDERIQUE JAUREGUI RENAUND, de quien me ha brindado sus sabios conocimientos y apoyo incondicional.

Con gran aprecio a mi asesor ING. FRANCISCO RAÚL ORTIZ GONZÁLEZ, que con su gran conocimiento y su valiosa experiencia me brindo su ayuda en la realización de esta tesis.

A mi primo Dr. en Física MIGUEL ÁNGEL OLIVARES ROBLES, de quien admiro y respeto como ejemplo en el camino del conocimiento.

CONTENIDO

Página

I

INTRODUCCIÓN

La práctica de la ingeniería siempre ha estado relacionada con la evolución de las nuevas tecnologías, ya sea para el desarrollo del área eléctrica, industrial, mecánica, etc. Los ingenieros han explotado nuevos sistemas y conceptos que han ampliado sus posibilidades, hoy en día, la tecnología de las computadoras ha proporcionado un sinnúmero de nuevas herramientas poderosas que tienen un profundo efecto en la profesión.

A lo largo del tiempo la computadora se ha hecho una herramienta cada vez más importante en la vida del ser humano, en aplicaciones cada vez más complejas y tareas menos comunes que tenia al principio de su invención. La computadora tiene aplicaciones en problemas matemáticos, administrativos, entretenimiento y comunicación, meteorológicos, espaciales, de inteligencia artificial (sistemas expertos, redes neuronales, robótica, etc.) entre otras áreas, debido al gran alcance que tiene la computadora, en esta tesis se utilizará como herramienta principal.

Sin embargo la mayoría de los usuarios consideran a este dispositivo electromecánico como algo referente a un monitor, un gabinete en cuyo interior se encuentran: CPU, memoria, tarjeta madre, tarjeta de video, entre otros, teclado e impresora; y se la ha hecho poco énfasis a aplicaciones del manejo de sistemas físicos, es decir, al empleo de dispositivos o componentes de manera controlada desde cualquier computador.

Conseguir el mejor provecho de la computadora para que realice tareas que sean repetitivas, peligrosas, precisas, y de control de equipos remotos entre otros, a través de radiofrecuencia o Internet. De esa manera el ser humano aumenta su calidad de vida, de trabajo, y también favorece a mejorar las principales características que tiene el ser humano (creatividad, imaginación, innovación, inventiva, etc.).

A continuación se describirán los aspectos principales en los que se encuentran comprendidos los capítulos de este trabajo:

Inicialmente tendrá la finalidad de describir históricamente la operación de equipos mecánicos como son la Pascalina, la Máquina de Babbage, así como los principios de la programación referente a la Máquina de Turín, hasta abarcar el Modelo de von Neumann.

Posteriormente se describirá el desarrollo de los diferentes tipos y clasificación de computadoras, haciendo un énfasis del Hardware de las PC`s y su arquitectura enfocado al diseño de los puertos, y aplicaciones.

Seguidamente se detallará los antecedentes de desarrollo y aplicaciones del lenguaje de programación "C" considerando funciones de salida, entrada, de decisión, repetición así como arreglos, cadenas, apuntadores y el diseño de librarías.

A continuación se tratará todo lo relacionado a los experimentos eléctricos, como la carga estática, leyes eléctricas, fundamentos, campos magnéticos y eléctricos, así como componentes electrónicos (diodo, transistor) y componentes digitales y de diseño lógico (compuertas lógicas, buffers, simplificación de circuitos, etc.).

Y por último se describirá la aplicación que existe entre un software diseñado, con el lenguaje de programación "C", aplicado al funcionamiento y control de velocidad de motores eléctricos, por medio de puertos de computadora (paralelo, serie, USB). Así como algunas aplicaciones en Visual Basic, orientado a lo mismo.

CAPÍTULO 1

EQUIPO DE CÓMPUTO

1.1. ANTECEDENTES

Los dedos de la mano se consideran una de las primeras herramientas de cálculo, y también gracias a los 10 dedos (5 izquierda y 5 derecha) de la mano, se creó el sistema decimal es decir, que contiene diez dígitos (Dígito del latín *digitus*, o sea dedo).

Los dedos de la mano fueron utilizados para manejar cantidades muy pequeñas, puesto que a cada dedo le correspondía un elemento a contar, y únicamente el ser humano podía contar hasta diez, el método era efectivo pero no muy práctico, ya que si la cantidad que deseaba manejar superaba en número los dedos de su mano, su instrumento de conteo quedaba sin concluir.

Cuando las sociedades primitivas fueron evolucionando y el hombre comenzó a tener un mayor número de propiedades, tuvo la necesidad de crear nuevas formas de contar, así inventó otro método para guardar información que curiosamente fue una adaptación del primero, es decir, como el número de dedos de la mano se limitaba a diez, optó por representar cada dedo por medio de una línea pintada en un árbol, piedra, vara o piel seca de animal. Para contar con esta nueva herramienta, el hombre colocaba conjuntos de cuatro líneas atravesadas por otra, para así formar "manos" de información.

Otra de las formas de contar y guardar información más exitosa, fue hacer nudos en cuerdas, cada nudo representaba un objeto, la ventaja de esta herramienta era que en el momento de hacer trueques podía fácilmente colocar más nudos a sus cuerdas si adquirían pertenencias, o bien deshacer nudos si disminuían.

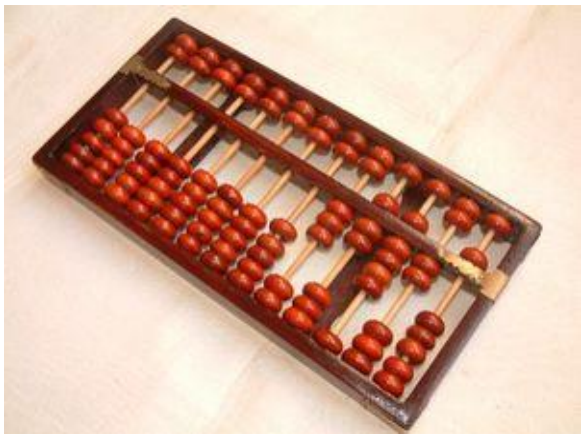


FIGURA 1.1. ÁBACO.

Sin embargo, el primer dispositivo formal es el ábaco, hace más de 2,500 años, su origen hoy en día se tiende a pensar que se encontraba en China (siglo V a. C.).

A finales de la Edad Media los mongoles introdujeron el ábaco en Rusia, que provenía de los chinos y los tártaros y que todavía hoy se utiliza en el pequeño comercio. En China y Japón , también hoy muy a menudo lo utilizan los hombres de negocios y contables, además los usuarios expertos son capaces de hacer operaciones más rápidas que con una calculadora electrónica.

El inventor y pintor **Leonardo da Vinci** (1452-1519), trazó las ideas para una sumadora mecánica. 150 años después, el filósofo y matemático francés **Blas Pascal** (1623-1662), por 1642, inventa la primera sumadora mecánica. A esta máquina se le conoce como **la Pascalina** en honor a su inventor y utilizaba engranes interconectados que representaban los números 0 a 9. Una nota importante de este inventor es que solo contaba con 19 años cuando realizó su invento.

FIGURA 1.2. LA PASCALINA.

Posteriormente, **Leibniz Gottfried Wilhem** (1646-1716), también conocido como el barón Gottfried Wilhem von Leibniz, filósofo, matemático y estadista alemán, perfeccionó el diseño de Pascal y creó un aparato de cálculo que podría llevar cabo las cuatro operaciones aritméticas básicas (suma, resta, multiplicación y división).

En 1804, **Joseph Marie Jacquard** perfeccionó el telar automático, mediante el uso de perforaciones en una serie de tarjetas con la finalidad de poder controlar el tejido de las telas. El telar leía el diseño codificado en las tarjetas y tejía la tela adecuadamente. Cabe mencionar que estas tarjetas perforadas y codificadas fueron las antecesoras de las tarjetas perforadas que se utilizaron en las primeras computadoras modernas.

En la década de 1820, el inglés **Charles Babbage**, (1792-1871), inventor y matemático británico, comenzó a desarrollar su **máquina diferencial**, un aparato que podía realizar cálculos matemáticos sencillos. Aunque Babbage empezó a construir esta máquina, no pudo terminarla por falta de fondos económicos. En la década de 1830, empezó a desarrollar otra nueva máquina llamada **máquina analítica**, que fue concebida para llevar acabo cálculos más complicados, pero este aparato tampoco se construyó.

El libro de Babbage “Tratado de Economía de Máquinas y de Manufacturas (1832)”, inició el campo del estudio conocido actualmente como investigación operativa. Sin embargo en el siglo XX, específicamente en el año 1991, unos científicos británicos, que siguieron los dibujos y las especificaciones detalladas de Babbage, construyeron esa máquina diferencial, la cual funcionaba a la perfección y hacia cálculos exactos con 31 dígitos, lo que demostraba que el diseño de Babbage era correcto.

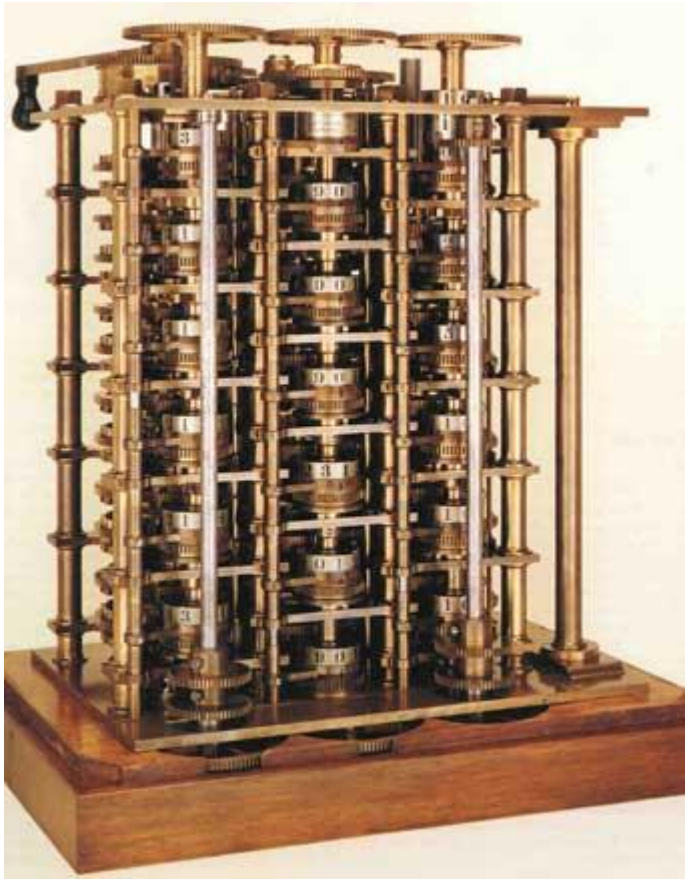


FIGURA 1.3. MÁQUINA DIFERENCIAL DE BABBAGE

En 1854, el matemático inglés **George Boole** crea el álgebra booleana y establece las bases de la teoría de la información.

El Álgebra de Boole (también llamada Retículas booleanas) en informática y matemática, son estructuras algebraicas que "capturan la esencia" de las operaciones lógicas Y, O y NO, así como el conjunto de operaciones unión, intersección y complemento.

George Boole, fue el primero en definirla como parte de un sistema lógico a mediados del siglo XIX. Específicamente, el álgebra de Boole fue un intento de utilizar las técnicas algebraicas para tratar expresiones de la lógica proposicional. En la actualidad, el álgebra de Boole se aplica de forma generalizada en el ámbito del diseño de la electrónica digital.

Los operadores del álgebra de Boole pueden representarse de varias formas. A menudo se representan simplemente como AND (Y), OR (O) y NOT (INVERSOR). También se emplean la OR exclusiva y su negadas NAND, NOR.

Durante la década de 1880, el estadístico estadounidense **Herman Hollerith**, concibió la idea de utilizar tarjetas perforadas, similares a las de Jacquard, para procesar datos. Hollerith consiguió compilar la información estadística destinada al censo de población de 1890, de los Estados Unidos de América (EUA), mediante la utilización de un sistema que hacía pasar tarjetas perforadas sobre contactos eléctricos, cabe mencionar que en este censo, se capturó en aproximadamente 3 años, y no en los 11 años que la entidad había estimado inicialmente, ahorrándose 5`000,000 de dólares (USD).

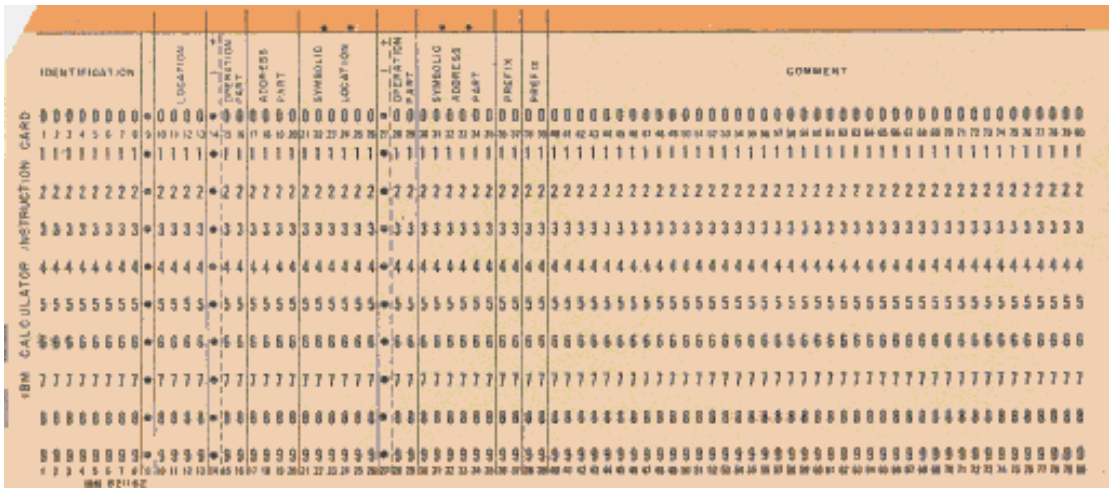


FIGURA 1.4. TARJETA PERFORADA

La **MARK I** en 1944, fue el primera computadora **electromecánica** construida en la Universidad de Harvard por el matemático **Howard Aiken,** el diseñó de la máquina fue considerada la primera computadora digital, por que trabajaba con estados lógicos. No obstante era un rudimentario modelo, construido con partes mecánicas en la que la secuencia de instrucciones para le resolución de problemas, debía ser alimentada a cada paso mediante un rollo de papel perforado.

Tenía 760,000 ruedas y 800 kilómetros de cable y se basaba en la Máquina Analítica de Babbage.

Esta computadora empleaba señales electromagnéticas para mover las partes mecánicas. También era lenta (tomaba de 3 a 5 segundos por cálculo) e inflexible (la secuencia de cálculos no se podía cambiar); pero ejecutaba operaciones matemáticas básicas y cálculos complejos de ecuaciones, sobre el movimiento parabólico de proyectiles.

Funcionaba con relés, se programaba con interruptores y leía los datos con cintas de papel perforado.

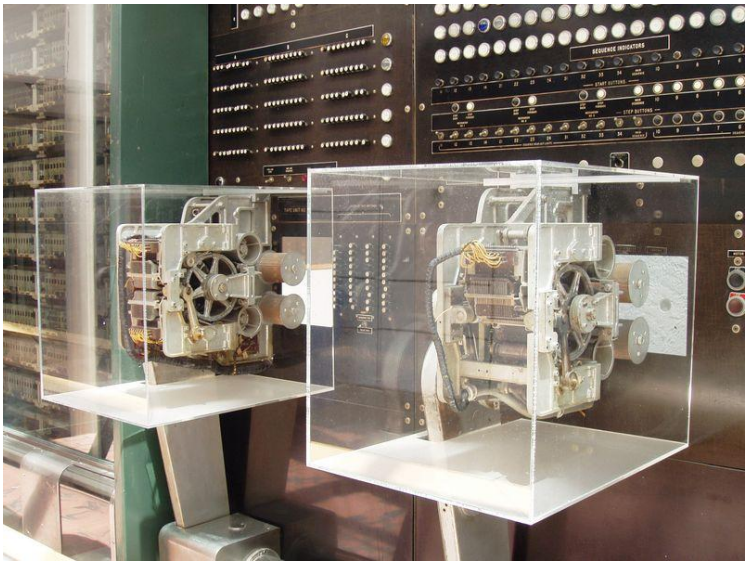


FIGURA 1.5. LA MARK I.

La **ENIAC,** es un acrónimo inglés de Electronic Numerical Integrator And Computer (Computador e Integrador Numérico Electrónico), utilizada por el Laboratorio de Investigación Balística del Ejército de los Estados Unidos de América.

Máquina gigantesca que ha sido la primera computadora **electrónica** de **propósito general** (a excepción del Colossus, que fue usado para descifrar código alemán durante la Segunda Guerra Mundial y destruida por la finalización del conflicto bélico, tras su uso para evitar dejar pruebas, siendo recientemente restaurada para un museo británico), la ENIAC era totalmente digital, es decir, que ejecutaba sus procesos y operaciones mediante instrucciones en lenguaje máquina, a diferencia de otras computadoras contemporáneas de procesos analógicos. Y Fue presentada en público el 15 de febrero de 1946.

La ENIAC fue construida en la Universidad de Pennsylvania por John Presper Eckert y John William Mauchly, ocupaba una superficie de 167 m². Físicamente, esta computadora tenía 17,468 tubos de vacío, 7,200 diodos de cristal, 1,500 relés, 70,000 resistencias, 10,000 condensadores. Pesaba 27 ton, medía 2.4 m x 0.9 m x 30 m; utilizaba 1,500 conmutadores electromagnéticos y relés; requería la operación manual de unos 6,000 interruptores, y su programa o software, cuando requería modificaciones, tardaba semanas de instalación manual.

Otra de sus características era que elevaba la temperatura del local a 50 ºC. Para efectuar las diferentes operaciones era preciso cambiar, conectar y reconectar los cables como se hacía, en esa época, en las centrales telefónicas, de allí el concepto. Este trabajo podía demorar varios días dependiendo del cálculo a realizar. A las 23:45 hrs. del 2 de octubre de 1955, la ENIAC fue desactivada para siempre.

FIGURA 1.6. ENIAC.

Encargados de su funcionamiento de la ENIAC

Aunque resulta frecuente encontrar en numerosos textos el nombre de los creadores de la ENIAC, John Presper Eckert y John William Mauchly, no lo es tanto encontrar la referencia de quienes se encargaron de hacer funcionar la computadora. En la descripción del puesto de trabajo se decía: "Requiere esfuerzo, creatividad mental, espíritu innovador y un alto grado de paciencia" ya que, por cierto, la ENIAC no tenía manual de programación.

Quienes se encargaron de que la computadora funcionara fueron seis mujeres: Kay Antonelli (1921-2006), Jean Bartik (1924-), Betty Holberton (1917-2001), Marlyn Meltzer, francés Spence (1922-) y Ruth Teitelbaum (1924-86). Sus nombres fueron ocultados durante años y han sido recuperados recientemente para la historia de la computación.

Ellas desarrollaron los primeros programas de software de la primera computadora electrónica, y crearon el campo de la programación. A mediados de los años cuarenta del siglo XX, eran las únicas programadoras de computadoras de propósito general en el mundo. Fueron así las maestras de la primera generación de programadores digitales.

La historia de estas mujeres ha estado invisibilizada hasta que en 1986, una estudiante de Harvard, Kathryn Kleiman, descubrió la historia de estas mujeres al realizar una investigación sobre el papel de las mujeres en la computación. La información fue difundida por Tom Petsinger, que realizó un reportaje reconociendo el trabajo de las mujeres de la ENIAC para Wall Street Journal.

Ellas, junto a la primera programadora en la historia del ordenador Ada Lovelace también conocida como Ada Byron, son claves en el desarrollo de la informática.

La computadora podía calcular trayectorias de proyectiles, lo cual fue el objetivo primario al construirla. En 1.5 segundos era posible calcular la potencia 5,000 de un número de hasta 5 cifras.

La ENIAC podía resolver 5,000 sumas y 360 multiplicaciones en 1 segundo. Pero entre las anécdotas estaba la poco promisoria cifra de un tiempo de rotura de 1 hora.

En 1944, el matemático húngaro-estadounidense, de ascendencia judía **John von Neumann** fue pionero de la computadora digital moderna, y de la aplicación de la teoría operadora a la mecánica cuántica. Publicó un artículo acerca del almacenamiento de programas. El concepto de **programa almacenado,** permitió la lectura de un programa dentro de la memoria de la computadora, y después la ejecución de las instrucciones del mismo, sin tener que volverlas a escribir.

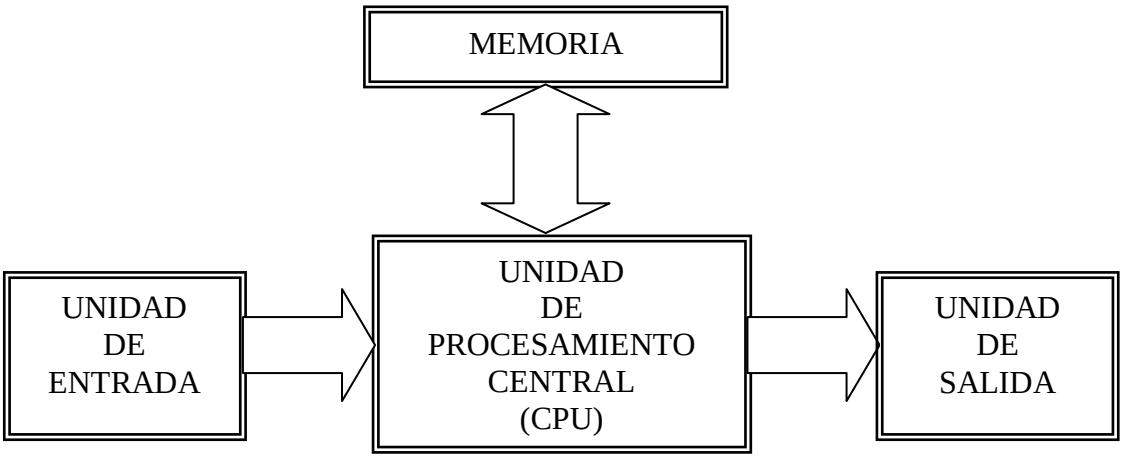


FIGURA 1.7. MODELO DE JOHN VON NEUMANN.

La primera computadora digital en usar el concepto de **programa almacenado,** gracias a John von Neumann fue la llamada **EDVAC** (Electronic Discrete-Variable Automatic Computer, es decir Computadora Automática Electrónica de Variable Discreta), desarrollada por los citados Eckert, Mauchly y el recién integrado Von Neumann. El cual firmaron un contrato en abril de 1946, para construirla con un presupuesto inicial de 100,000 USD, y al finalizar el proyecto tuvo un costo de 500,000 (USD), igual que la ENIAC.

Los programas almacenados dieron a las computadoras flexibilidad y confiabilidad, haciéndolas más rápidas y menos sujetas a errores que los programas mecánicos.

Así como la ENIAC, la EDVAC, fue construida por el laboratorio de investigación de balística de Estados Unidos de la universidad de Pensilvana.

La EDVAC, poseía físicamente casi 6,000 tubos de vacío y 12,000 diodos. Consumía 56 Kilowatts de potencia. Cubría 45.5 m² de superficie y pesaba 7,850 Kg. El personal operativo consistía de treinta personas para cada turno de ocho horas.

Fue entregada al laboratorio militar en agosto de 1949, y después de varios ajustes, comenzó a operar hasta 1951. En 1960, operaba por más de 20 horas diarias, con lapsos sin error de 8 horas en promedio.

Recibió varias actualizaciones, incluyendo un dispositivo de entrada/salida de tarjetas perforadas en 1953, memoria adicional en un tambor magnético en 1954 y una unidad de aritmética de punto flotante en 1958.

La EDVAC operó hasta 1961, cuando fue reemplazada por BRLESC. En su vida demostró ser altamente confiable y productiva.

En 1951, **UNIVAC I** fue la primera computadora **comercial**. Los doctores Mauchly y Eckert fundaron la compañía Universal Computer (UNIVAC), y su primer producto fue esta máquina. El primer cliente fue la oficina del censo de la Unión Americana (EUA).

En 1953, IBM (International Business Machines) construye la **701,** basada también en válvulas al vacío, y para introducir los datos en estos equipos, empleaban el concepto de tarjetas perforadas, que había sido inventada en los años de la revolución industrial (finales del siglo XVIII), por el francés Jacquard y perfeccionado por el estadounidense Herman Hollerith en 1890. La IBM 701, fue la primera de una larga serie de computadoras de esta compañía, que luego se convertiría en la número uno por su volumen de ventas. Y consecutivamente a finales de 1958, fue creada la primera computadora de IBM basada en **transistores,** llamada 7090.

FIGURA 1.8. IBM 701.

Posteriormente la empresa IBM en 1975, sentaron las bases del concepto de **"máquina modular"**, para ello se construyó el modelo **Altair**; esta pequeña computadora se diseñó con base a una arquitectura abierta, mediante ranuras o "slots" para conectar aditamentos y periféricos de otras marcas. Entre otros diseños de la época de los años 70`s del siglo XX, se aportaron conceptos tecnológicos en lo que descansaría la revolución del estándar PC (Personal Computer, Computadora Personal), de IBM que en 1981, penetró al mercado popular, y actualmente sigue en auge.

FIGURA 1.9. MODELO ALTAIR

En su propuesta, IBM quiso aprovechar la dinámica del mercado, y reunir en torno a su proyecto de fabricantes y tecnologías ya existentes, a fin de impulsar juntos una plataforma y establecer de manera definitiva un estándar de arquitectura abierta. Para ello, entre otras medidas, incluyó en su primera propuesta un microprocesador de Intel (8088); y también contrató de manera externa lenguajes y sistemas operativos (Microsoft).

En la actualidad ya es muy difícil precisar el término "compatible" con IBM, debido a que las diferencias que originalmente llegaron a existir entre marcas, han desaparecido conforme el desarrollo de las nuevas generaciones de computadoras PC, enriqueciendo incluso al propio estándar de IBM. De hecho, los conceptos de compatible o clón prácticamente han dejado de usarse, para hablar simplemente de computadoras PC, en contraste con otros estándares como Macintosh o Silicon Graphics.

1.2. MICROPROCESADOR

El microprocesador o micro, es un circuito integrado que contiene todos los elementos de una "Unidad Central de Procesamiento" o CPU (Central Processing Unit). Es el componente en una computadora digital, que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora. Los CPU proporcionan la característica fundamental de la computadora digital, la programabilidad, y son uno de los componentes necesarios encontrados en las computadoras de cualquier tiempo, junto con el almacenamiento primario, y los dispositivos de entrada/salida. Se conoce como microprocesador el CPU, que es manufacturado con circuitos integrados. Desde mediados de los años 70`s del siglo XX, los microprocesadores de un solo chip han reemplazado casi totalmente todos los tipos de CPU, y hoy en día, el término "CPU" es aplicado usualmente a todos los microprocesadores.

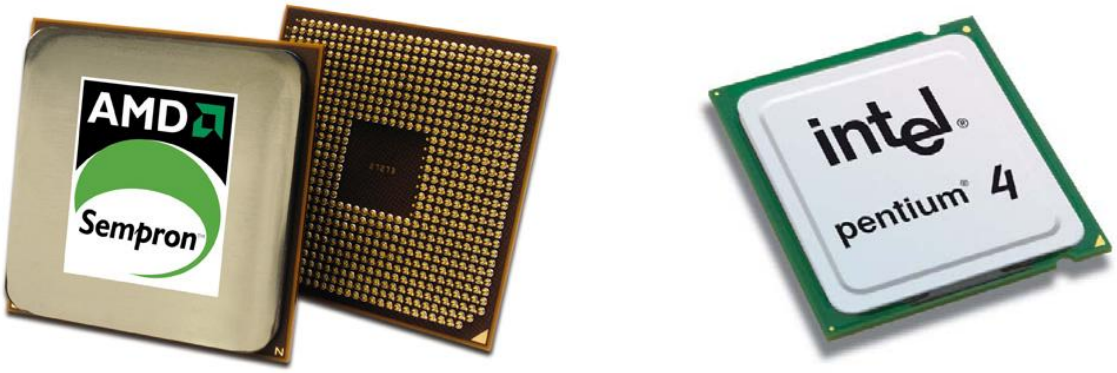


FIGURA 1.10. MICROPROCESADORES

Un microprocesador, realiza operaciones matemáticas y lógicas, para el cumplimiento de una tarea determinada, en función de una serie de instrucciones suministradas por un programa externo, y parte de las cuales se almacenan internamente en el mismo circuito.

1.2.1. PRINCIPALES COMPONENTES DE UN MICROPROCESADOR

1.- LA UNIDAD ARITMÉTICA LÓGICA (ALU: Arithmetic and Logia Unit).- Es la responsable de todas las operaciones aritméticas y lógicas. Las funciones lógicas en general incluyen NOT, AND, OR. La ALU no efectúa multiplicación o división en forma directa, sino que estas funciones, se llevan a cabo mediante una secuencia de instrucciones de suma, resta y desplazamiento, por medio del lenguaje binario.

2.- REGISTROS.- Es donde se almacenan temporalmente los datos y es extraída por el ALU, posteriormente enviados a al RAM. Los principales registros del microprocesador son:

a) **Acumuladores.-** Es un registro de uso general. La mayoría de las operaciones dentro del procesador se efectúa sobre el contenido de estos registros, o resultan afectadas por él. En los microprocesadores varían en el número de acumuladores que tiene.
b) **Registro índice**.- Se usan para almacenar información, sobre las direcciones.
c) **Contador de programa**.- Es una lista de instrucciones al procesador. Estas instrucciones ordenadas se almacenan dentro de una sección de memoria, y se extrae una a la vez, para su ejecución.
d) **Registro de instrucciones.-** Cada instrucción se lleva, a su vez a un registro de instrucción para su ejecución. Este registro se conecta a la unidad de control, la cual produce una secuencia apropiada de señales de control para implementar esta instrucción.
e) **Registro de status del microprocesador.-** Los bits individuales de este registro se pone uno y a cero, para indicar ciertos aspectos del estado de la ALU (Unidad Aritmética Lógica), y en si del microprocesador..
f) **Apuntador de pila**.- Es un registro de propósito especial. Su función es almacenar una dirección, que apunta hacia una posición de memoria.

3.- LA UNIDAD DE CONTROL Y DECODIFICACIÓN.- Esta unidad es el corazón del microprocesador. Tiene la función de extraer en forma secuencial instrucciones de la memoria y luego ejecutarlas. Unido a la unidad se encuentra un **generador de reloj**, que en general utiliza un oscilador de cristal para producir una señal de reloj muy precisa, y como resultado sincronizar perfectamente la ejecución de todas las operaciones.

4.- BUS DE DIRECCIÓN Y DATOS.- Los **buffers** forman parte del bus que comunican al microprocesador, con los componentes externos dentro del chip. Dentro del microprocesador las resistencias son altas y las capacidades bajas, lo cual permite que se usen señales de muy baja potencia. Cuando las señales actúan sobre cargas externas, con sus capacidades asociadas, se necesita más potencia, los **buffers** proporcionan esta potencia adicional.

1.2.2. ORIGEN DEL MICROPROCESADOR

Esto ocurrió en 1971, a solicitud de la compañía japonesa Busicom, fabricante de calculadoras electrónicas. Busicom encomendó a **Intel** (entonces un fabricante modesto de circuitos integrados) el desarrollo de las unidades de proceso central, de 12 modelos distintos de calculadoras que deseaba lanzar al mercado. Poco después de comprometerse a hacer el trabajo, los ingenieros de Intel se dieron cuenta que no tendrían tiempo suficiente para producir 12 integrados distintos, así que decidieron diseñar un circuito integrado central, en las que se encontraban todas las funciones de cálculos deseadas; y las

particularidades de cada modelo, se colocaron en una memoria ROM (Read-Only Memory, que significa "memoria de sólo lectura independiente").

De esta manera, sin que los ingenieros de Intel advirtieran aún las potencialidades de su solución, diseñaron un dispositivo de aplicaciones amplias, cuyas particularidades operacionales dependían del programa almacenado en la ROM externa; este componente básico se convirtió en el primer microprocesador, y fue comercializado por Intel, con la matrícula **4004** presentado al mercado el 15 de noviembre de 1971. Era un dispositivo de 4 bits de capacidad, y con una la velocidad del reloj de 100 KHz (Kilohertz), y un máximo de 640 bytes de memoria.

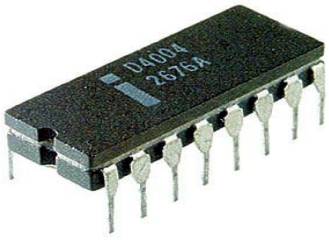


FIGURA 1.11. PRIMER MICROPROCESADOR DE INTEL, 4004

Texas Instruments, al igual que Intel en 1971, comparte el mérito de la invención casi simultánea del microprocesador, quien también obtuvo la primera patente del microprocesador, de un solo chip (inventado por Gary Boone) en 1973. Texas Instruments, es una empresa norteamericana con sede en Dallas Texas, Estados Unidos de América (EUA) que desarrolla y comercializa semiconductores y tecnología para computadoras. Texas Instruments es el tercer fabricante de semiconductores del mundo, tras Intel y Samsung, y es el mayor suministrador de circuitos integrados para teléfonos móviles

Texas Instruments fue fundada por Cecil H. Green, J. Erik Jonsson, Eugene McDermott y Patrick E. Haggerty a principio de los años 40´s del siglo XX.

La serie 7400 de circuitos integrados, basados en la tecnología TTL (Lógica Transistor a Transistor), desarrollada por Texas Instruments en los años de 1960, popularizó el uso de circuitos integrados en las computadoras y continúa siendo ampliamente utilizada hoy en día. Otros inventos de Texas Instruments fueron la calculadora de bolsillo en 1967.

1.2.3. LEY DE MOORE

El Dr. Gordon Moore, uno de los fundadores del Intel Corporation, formuló en el año de 1965, una ley que se ha venido a conocer como la " Ley de Moore". La citada ley nos viene a decir, que el número de transistores contenido en un microprocesador se dobla más o menos de cada 18 meses. Esta afirmación que en principio estaba destinada a los dispositivos de memoria, pero también los microprocesadores han cumplido la ley. Una ley que significa para el usuario que cada 18 meses, de forma continua pueda disfrutar de una mejor tecnología, algo que se ha venido cumpliendo durante los últimos 30 años del siglo anterior, y se espera siga vigente en los próximos 15 o 20 años del nuevo milenio (siglo XXI).

De modo que el usuario puede disponer de mejores equipos, aunque también signifique la necesidad de cambiar de equipo cada poco tiempo, algo que no todo el mundo se puede

permitir, y eso que el precio aumenta de forma obsoleta pero no relativa, puesto que la relación MIPS(Millones de Instrucciones por Segundo)-dinero está decreciendo a velocidad vertiginosa. Algo que sin embargo no sucede con la industria del automóvil, ya que la potencia de los coches no se ha multiplicado de la misma forma que los precios, en cualquier caso, queda claro que en los próximos años nos espera una auténtica revolución en lo que a rendimiento de los procesadores se refiere, como ya predijera Moore hace más de 30 años.

1.2.4. FUTURO DE LOS MICROPROCESADORES

El último paso conocido ha sido la implementación de la nueva arquitectura de 0.25 micras, que viene a sustituir de forma rotunda la empleada hasta el momento, de 0.35 micras en los últimos modelos de procesador. Esto va a significar varias cosas en un futuro no muy lejano, para empezar la velocidad se incrementará una medida del 33% con respecto a la generación del anterior, es decir, el mismo procesador usando esta nueva tecnología puede ir un 33% más rápido que el anterior. Para hacernos una idea de este tamaño de tecnología, el valor de 0.25 micras son unas 400 veces más pequeños que un cabello de cualquier persona, y este tamaño es el que tienen los transistores que componen al microprocesador.

El transistor, como muchos sabemos, permite el paso de la corriente eléctrica, de modo que en función de en qué transistores halla corriente, la computadora realiza las cosas (esto es una simplificación de la realidad pero se ajusta a ella). Dicha corriente eléctrica circula entre dos puntos de modo que cuanto menor sea esta distancia, más cantidad de veces podrá pasar, pues el tiempo es menor.

Aunque estamos hablando de millonésimas de segundo, tener en cuenta que un procesador está trabajando continuamente, de modo que ese tiempo que parece insignificante, cuando es sumado a lo largo de las miles de millones de instrucciones que realizar, nos puede dar una cantidad de tiempo importante. De modo que la tecnología que se utilice puede dar resultados totalmente distintos, incluso utilizando el mismo microprocesador.

Dicen que es natural en el ser humano querer mirar constantemente hacia el futuro, buscando información de hacia donde vamos, en lugar de en donde hemos estado. Por ello, no podemos menos que asombrarnos, de las previsiones que los científicos barajan para dentro de unos 15 años. Según el Dr. Albert Yu, vicepresidente de Intel y responsable del desarrollo de los procesadores desde el año 1984, para el año 2011, utilizaremos procesadores cuyo reloj ira a una velocidad de 10 GHz (10,000 MHz) contendrán mil millones de transistores y será capaz de procesar cerca de 100 mil millones de instrucciones por segundo. Un futuro prometedor, permitirá realizar tareas nunca antes pensadas.

1.3. TARJETA MADRE

Es la plataforma sobre la que se construye una computadora: la tarjeta madre (también conocida como tarjeta principal, placa base o simplemente motherboard). En esta placa de

circuito impreso, destaca la arquitectura (abierta) de la máquina, recordando que las computadoras modernas fueron diseñadas para ser construidas en forma modular. Así, en esta placa se alojan el microprocesador, los circuitos que soportan el trabajo de este dispositivo, y los puertos de comunicación con el exterior.

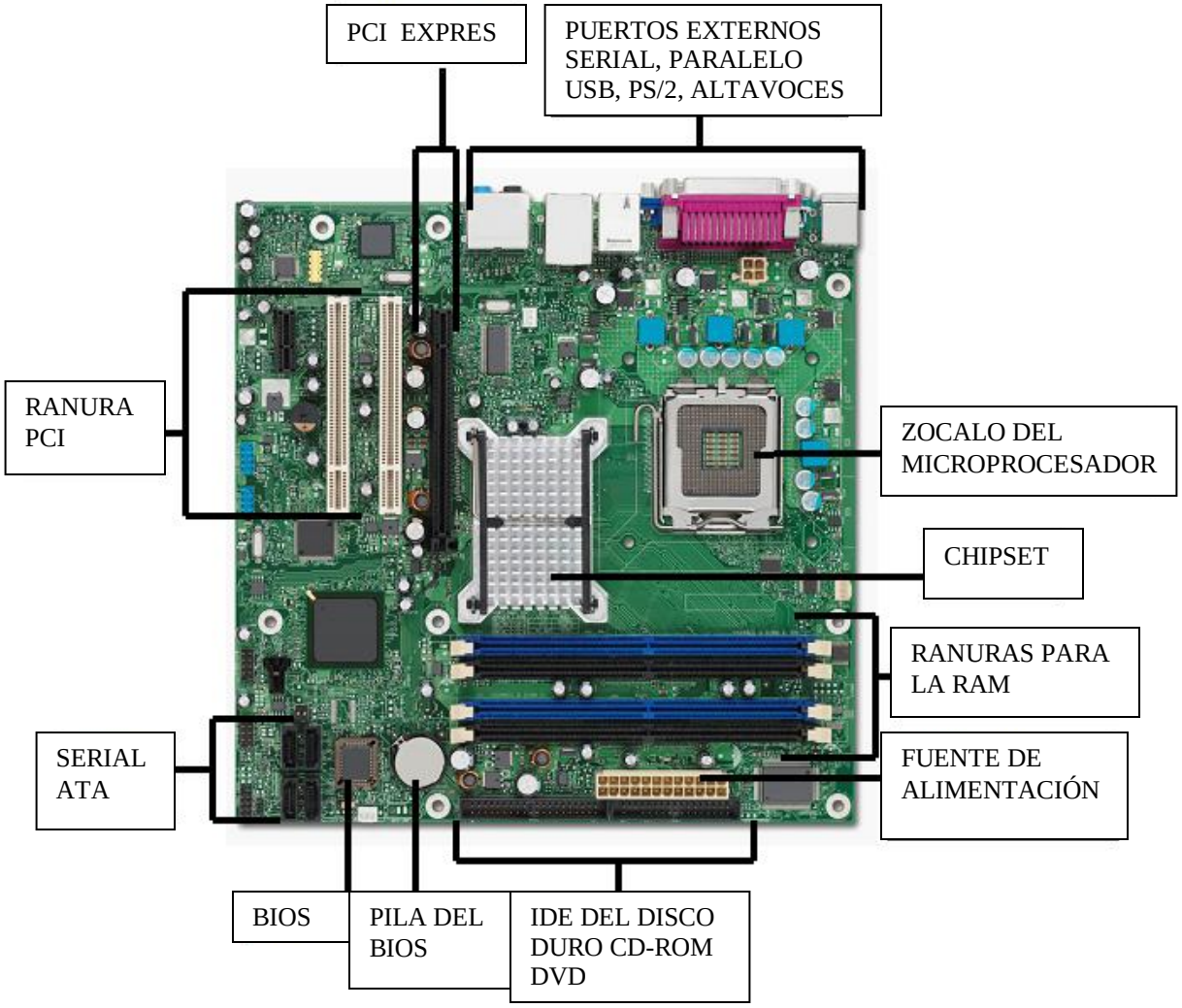


FIGURA 1.12. TARJETA MADRE.

1.3.1. ZÓCALO

Con respecto a la tarjeta madre, es el manejo de las comunicaciones desde y hacia el microprocesador. Por lo tanto, para que se dé esta comunicación, debe de existir un medio de interconexión entre ambos dispositivos; este medio físico es el zócalo o socket.

El zócalo o socket es una matriz de pequeños agujeros ubicados en una placa madre, es la base donde encajan, sin dificultad, los pines de un microprocesador. Esta matriz permite la conexión entre el microprocesador y el resto del equipo. En las primeras computadoras personales el microprocesador, venía directamente soldado a la placa base, pero la aparición de una amplia gama de microprocesadores llevó a la creación de los zócalos.

En general cada familia de microprocesadores, requiere un tipo distinto de zócalo, ya que existen diferencias en el número de pines, su disposición geométrica y la interconexión requerida con los componentes de la placa base. Por tanto, no es posible conectar un determinado microprocesador a una placa base diseñada para otro.

Donde: en la actualidad se utilizan cinco tipos de zócalo para distintos tipos de microprocesadores.

1. El socket 370, para los Pentium III, Celeron y VIAC3.
2. El soceket 462, para el Duron y el Athlon de AMD.
3. El soceket478, para el Pentium 4 y los nuevos Celeron.
4. El soceket 754, para los micros AMD Athlon-64, Sempron
5. El soceket 940, para los microprocesadores AMD Athlon.64fx y Optaron.

1.3.2. CHIPSET

El chipset, es un conjunto de circuitos integrados diseñados para trabajar conjuntamente, y en apoyo al microprocesador.

El chipset se encarga de manejar todas las señales lógicas que van al microprocesador o salen de este dispositivo. Además también es responsable en una medida importante del rendimiento global del mismo.

Tradicionalmente, el chipset se divide en dos bloques: el "puente norte" (north-bridge), que tiene a su cargo la comunicación de la memoria RAM con el puerto AGP y con otros circuitos de alta velocidad; y el "puente sur" (south-bridge), que reduce la velocidad para comunicarse con las ranuras PCI, con la interfaz IDE, con los puertos I/O (Input-Output, entradas-salidas) etc.

1.3.3. MEMORIA RAM

La memoria RAM (Random Acces Memory: Memoria de Acceso Aleatorio) es el almacén temporal de datos del microprocesador; ahí toma y deposita la información numérica (instrucciones o datos de trabajo) que precisa para sus operaciones; los datos sólo se mantienen mientras la computadora esté alimentada de energía eléctrica, es por ello se que pierden la información cuando el equipo se apaga. Han existido y existen diferentes tipos de memoria RAM las principales son las siguientes:

A) DIP (Dual in line package, Paquete en Línea Doble).-Es una forma de encapsulamiento común en la construcción de circuitos integrados. La forma consiste en un bloque con dos hileras paralelas de pines, la cantidad de éstos depende de cada circuito

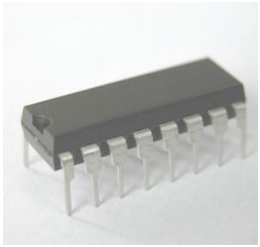


FIGURA 1.13. MEMORIA TIPO DIP.

B) SIPP (Single In line Pin Package, Paquete de Pines en Línea Simple).-Consiste en un circuito impreso (también llamado módulo) en el que se montan varios chips de memoria RAM, con una disposición de pines correlativa (de ahí su nombre). Tiene un total de 30 pines a lo largo del borde del circuito, que encajan con las ranuras o bancos de conexión de memoria de la placa base del ordenador, y proporcionan 4 bits por módulo. Se usó en sistemas 80286

FIGURA 1.14. MEMORIA TIPO SIPP

C) SIMM (Single In Line Memory Module, Modulo de Memoria en Línea Simple).-Es un tipo de encapsulado que consistente en una pequeña placa de circuito impreso que almacena chips de memoria, y que se inserta en un zócalo SIMM en la placa base o en la placa de memoria. Los contactos en ambas caras son redundantes, lo que es la mayor diferencia respecto de sus sucesores los DIMMs.

FIGURA 1.15. MEMORIA TIPO SIMM.

D) DIMM (Dual In line Memory Module, Modulo de Memoria en línea Doble).-Las memorias DIMM comenzaron a reemplazar a las SIMMs como el tipo predominante de memoria cuando los microprocesadores Intel Pentium dominaron el mercado. Son módulos de memoria RAM utilizados en ordenadores personales. Se trata de un pequeño circuito impreso que contiene chips de memoria y se conecta directamente en ranuras de la placa base.

FIGURA 1.16. MEMORIA TIPO DIMM.

E) DDR (Double Data Rate, Doble Tasa de Transferencia de Datos).-Memoria síncrona, envía los datos dos veces por cada ciclo de reloj. De este modo trabaja al doble de velocidad del bus del sistema, sin necesidad de aumentar la frecuencia de reloj. Se presenta en módulos DIMM de 184 contactos.

FIGURA 1.17. MEMORIA TIPO DDR.

F) DDR-2 (Double Data Rate, Doble Tasa de Transferencia de Datos-Dos) Las memorias DDR-2 son capaces de trabajar con 4 bits por ciclo, es decir 2 de ida y 2 de vuelta en un mismo ciclo mejorando sustancialmente el ancho de banda, bajo la misma frecuencia de una DDR tradicional, (si una DDR a 200 MHz reales entregaba 400 MHz nominales, la DDR-2 por esos mismos 200 MHz reales entrega 800 MHz nominales).

FIGURA 1.18. MEMORIA TIPO DDR2.

1.3.4. RANURAS DE EXPANSIÓN

Hemos insistido en que una de las razones principales del gran éxito de las computadoras personales, es su enorme capacidad de crecimiento, que a su vez descansa en las tarjetas auxiliares que se conectan en ranuras de expansión (slots) especiales.

1.3.4.1. Ranura AGP (Accelerated Graphics Port, Puerto de Gráficos Acelerado)

Esta ranura, especialmente diseñada para conectar la tarjeta de video, posee la suficiente velocidad y ancho de banda como para manejar el enorme flujo de datos que requieren las modernas aplicaciones multimedia; por ejemplo, los juegos o las películas en DVD (Disc Versátil Digital, Disco Versátil Digital).

El bus AGP cuenta con diferentes modos de funcionamiento.

- AGP 1X: velocidad 66 MHz con una tasa de transferencia de 264 MB/s y funcionando a un voltaje de 3,3V.
- AGP 2X: velocidad 133 MHz con una tasa de transferencia de 528 MB/s y funcionando a un voltaje de 3,3V.
- AGP 4X: velocidad 266 MHz con una tasa de transferencia de 1 GB/s y funcionando a un voltaje de 3,3 o 1,5V para adaptarse a los diseños de las tarjetas gráficas.
- AGP 8X: velocidad 533 MHz con una tasa de transferencia de 2 GB/s y funcionando a un voltaje de 0,7V o 1,5V.

1.3.4.2. Ranura PCI (Peripheral Component Interconnect, Interconexión de Componentes Periféricos)

Consiste en un bus de ordenador estándar para conectar dispositivos periféricos directamente a su placa base o tarjeta madre, por medio de una ranura de expansión para proporciona una adecuada velocidad de transferencia de datos sin grandes costos. Es un sistema normal, el bus PCI se usa para manejar tarjeta de video, los puertos I/O(Entradas-salidas), la tarjeta de sonido, el módem, la tarjeta de red, etc. Pero esta ranura ya se esta volviendo lenta (trabaja a una frecuencia máxima de "sólo" 33MHz), comparada con las velocidades de los microprocesadores, y tarjetas madre actuales; aun así, sigue siendo adecuada para muchas aplicaciones.

1.3.4.3. PCI-Express

Ranura PCI-Express, anteriormente conocido por las siglas 3GIO, 3ra. Generation I/O(entradas-salidas), es un nuevo desarrollo del bus PCI que usa los conceptos de programación y los estándares de comunicación existentes, pero se basa en un sistema de comunicación serie mucho más rápido. Este sistema es apoyado principalmente por Intel.

PCI-Express es una evolución de PCI, en la que se consigue aumentar el ancho de banda mediante el incremento de la frecuencia, llegando a ser 32 veces más rápido que el PCI 2.1. Su velocidad es mayor que PCI-Express, pero presenta el inconveniente de que al instalar más de un dispositivo la frecuencia base se reduce y pierde velocidad de transmisión.

La velocidad del PCI-Express permitirá reemplazar casi todos los demás buses, AGP y PCI. La idea de Intel es tener un solo controlador PCI-Express comunicándose con todos los dispositivos. Este conector es usado mayormente para conectar tarjetas graficas.

PCI-Express en 2006 es percibido como un estándar de las placas base para PC, especialmente para conectar tarjetas gráficas. Marcas como Ati Technologies y nVIDIA entre otras tienen tarjetas gráficas utilizan PCI-Express.

1.3.5. PUERTOS I/O (Input/Output, Entradas/Salidas)

Para comunicarse con elementos externos, la plataforma PC dispone de una amplia variedad de puertos de entrada y salida de datos. Se conocen genéricamente como "puertos I/O (Input/Output, Entradas/Salidas).

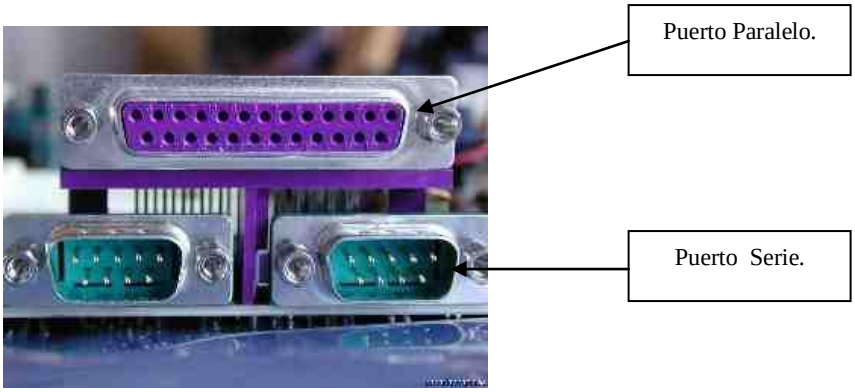


FIGURA 1.19. PUERTO PARALELO Y SERIE.

1.3.5.1. Puertos seriales

Estos permiten la comunicación con dispositivos de baja velocidad; por ejemplo, un módem externo, un ratón, un programador de EEPROM (Electrically Erasable and Programmable Read Only Memory, Memoria de Sólo Lectura Eléctricamente Programable y Borrable).

El puerto serie o serial, como su nombre lo indica, se caracteriza principalmente por que los datos viajan a través de un par de líneas de comunicación; una para el envío, y otra para el regreso (con su respectiva línea de "tierra"). Entonces, si dos terminales (cuatro terminales, si contamos los retornos de tierra) son suficientes para que los datos fluyan, no deja de ser raro que el puerto serie requiera de un conector de 25 terminales.

En realidad, el conector de 25 terminales se colocó únicamente para cumplir las especificaciones de la interfaz RS-232 (establecida antes de que se diseñara la IBM PC), pero de esas 25 líneas, muchas quedan sin conexión.

Ante esta situación, cuando aparecieron las máquinas AT (Alta Tecnología), las que usaban el microprocesador 386, también surgió un nuevo tipo de puerto serie. Este puerto, dotado con un conector de apenas 9 terminales, se sigue usando hasta nuestros días.

Los primeros puertos serie podían manejar comunicaciones de muy baja velocidad; por lo general, hasta 2,400 bits por segundo (bps). Pero conforme se fueron desarrollando nuevos circuitos de comunicaciones, esté parámetro aumentó considerablemente. Esto permite conectar dispositivos de velocidad relativamente alta; (en comparación con las primeras versiones del puerto serie); aun y cuando este conector siga siendo el más lento de los que se usan regularmente en la plataforma PC. Precisamente por esta razón, los puertos serie parece estar condenados a desaparecer; muchas tarjetas madre modernas, sólo cuentan con uno de ellos.

1.3.5.2. Puertos paralelos

La frecuencia máxima con la que los puertos paralelos y los puertos seriales transfieren datos, es de 8 MHz.

Los creadores de la PC decidieron colocar junto al puerto serie un puerto adicional, pero que fuera capaz de manejar más de un bit a la vez. Y así surge el puerto paralelo, que posee **ocho líneas dedicadas a la transmisión o recepción de datos**. Esto significa, en teoría, que puede manejar un byte de información para cada ciclo de reloj; por eso tiene un ancho de banda teórico de hasta 8 MB/s (millones de bytes por segundo), considerando la velocidad de trabajo máxima del bus ISA (Industry Standard Architecture , Arquitectura estándar industrial),

Por sus características, el puerto paralelo se uso, casi desde un principio para conectar una impresora. De hecho, originalmente se diseñó para comunicación unidireccional de la computadora con algún equipo periférico; casi no había modo de introducir información a través de dicho puerto. Pero conforme fue creciendo la cantidad de periféricos disponibles, los fabricantes de computadores se percataron de la conveniencia de contar con un puerto, que también permitiera la introducción de datos de alta velocidad; y para no tener que diseñar y colocar un puerto extra, modificaron las características operativas del puerto paralelo para convertirlo en bidireccional; o sea, para que pudiera enviar y recibir datos en paralelo.

De está adaptación, se derivó el surgimiento de los digitalizadores de imagen (escáneres), unidades de disco externas, impresoras de alta resolución y, en general, todos los

periféricos que necesitan de un puerto paralelo. Pero como la mayoría de computadoras contaban con dos puertos seriales y un solo puerto paralelo, el usuario tenía que desconectar un periférico para conectar otro que fuese ocupar. Fue así para crearon algunos conmutadores mecánicas y electrónicos, mediante los cuales el único puerto paralelo puede ser compartido por todos los dispositivos que el usuario requiera.

1.3.5.3. El puerto USB

A principios de la década de 1990, varías compañías líderes en el mundo de la informática decidieron diseñar un nuevo puerto de alta velocidad para la plataforma PC. Entre ellas, Intel figuraba como una de las principales patrocinadoras del proyecto. El resultado de sus esfuerzos, fue la creación del puerto USB (**Bus Serial Universal)**.

En abril del año 2000 fue liberada la **versión 2 del puerto USB**. Con esto, el ancho de la banda se ha ampliado hasta unos 480 Mbps (Millones de bits por segundo); esto permite un amplio espacio de maniobra, incluso con dispositivos que requieren de muchos recursos de comunicaciones.

1.3.5.4. El puerto Fire Wire

El estándar USB 1.0 ofrecía una velocidad de transmisión de datos que apenas llegaba a 12Mbps. Esto era insuficiente para aplicaciones muy demandantes; por ejemplo, para la captura de video digital en el disco de la PC. Ante este panorama, la empresa Apple Computer, cuyos sistemas Macintosh siempre han estado enfocados al manejo de aplicaciones gráficas intensivas, decidió crear un nuevo estándar de comunicaciones entre computadora y un periférico de alta resolución). Fruto de las investigaciones es el puerto IEEE-1394, mejor conocido por su nombre comercial: Fire Wire.

1.4. MEDIOS DE ALMACENAMIENTO

1.4.1. EL DISCO DURO

La primera unidad de disco duro que se tenga noticia, fue desarrollada por IBM. Se vendió bajo el nombre comercial de RAMAC-305, (Random Access Method of Accounting and Control, Método de Control y Contabilidad de Acceso Aleatorio).

Este medio de almacenamiento magnético estaba integrado por 50 discos metálicos de 60 centímetros de diámetro; apilado verticalmente montados en un gabinete de 2 metros de altura, 3 metros de ancho y 1 de profundidad.

FIGURA 1.20. DISCO METÁLICO DE 60 CM.

FIGURA 1.21. GABINETE DEL RAMAC-305.

Considerando tales dimensiones, se creía que en dicha unidad, almacenaba mucha información; pero la capacidad real de RAMAC-305, era de apenas 5 MB (MegaByte); aunque esto parece ridículo en nuestros días, en aquel entonces una aplicación promedio de aquellas fechas (alrededor de 1950), difícilmente excedía los 20 KB (KiloByte) de extensión, esto significa que la capacidad de esta unidad era muy amplia para su época.

Una Simple Caja Cerrada "EL HARD DISK (Disco Duro)"

Se llama disco duro o disco solido (en inglés hard disk, abreviado con frecuencia HD o HDD) al dispositivo encargado de almacenar información de forma permanente en una computadora. Y son unas reducidas cajas metálicas herméticamente cerradas del tamaño similar al de un libro pequeño.

Los discos duros generalmente utilizan un sistema de grabación magnética digital. En este tipo de disco encontramos dentro de la carcasa una serie de platos metálicos apilados girando a gran velocidad. Sobre estos platos se sitúan los cabezales encargados de leer o escribir los impulsos magnéticos. Hay distintos estándares a la hora de comunicar un disco duro con la computadora. Los más utilizados son Integrated Drive Electronics (IDE), SCSI, y SATA, este último estandarizado en el año 2004. Sus capacidades de almacenamiento va desde los 10 MB (MegaByte) hasta miles de GB (GigaByte).

FIGURA 1.22. DISCO DURO DE UNA PC.

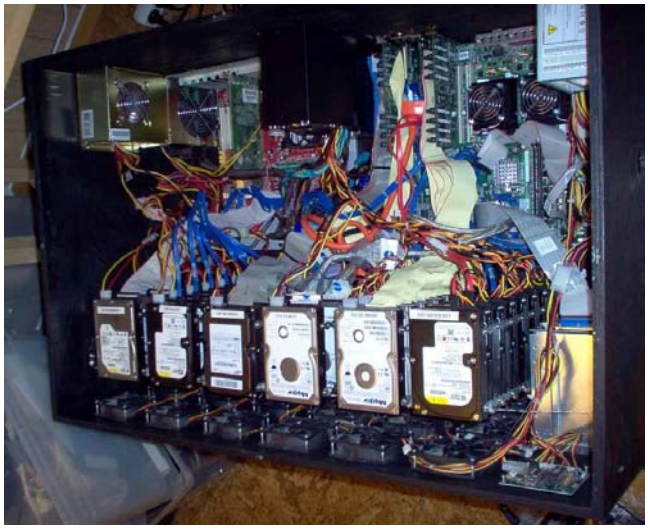


FIGURA 1.23. DISCOS DUROS DE UN SERVIDOR.

Dentro de un **disco** duro hay varios platos (entre 2 y 4), que son discos (de aluminio o cristal). Cada plato tiene dos caras, y es necesaria una **cabeza** de lectura/escritura para cada cara (no es una cabeza por plato, sino una por cara). Por tanto, hay 8 cabezas para leer 4 platos. Las cabezas de lectura/escritura nunca tocan el disco, sino que pasan muy cerca (hasta a 3 nanómetros). Si alguna llega a tocarlo, causaría muchos daños en el disco, debido a lo rápido que giran los platos (uno de 7,200 revoluciones por minuto se mueve a 120 Km/h en el borde).

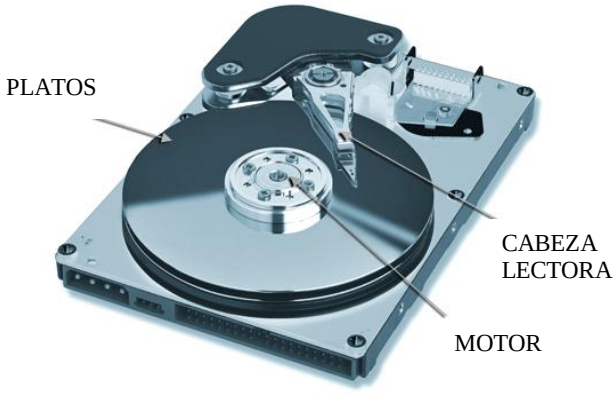


FIGURA 1.24. INTERIOR DEL DISCO DURO.

1.4.2. LOS DISQUETES

Los primeros disquetes que se fabricaron, poco tenían en común con los que ahora conocemos; su diámetro era de 8 pulgadas (20 centímetros), y podían almacenar hasta 80 KB de información; es poco en nuestros días, pero suficiente para las necesidades de la

época. En 1976, aparecieron los primeros disquetes de 5.25 pulgadas (13.5 centímetros), cuya capacidad total era de 160 KB.

Poco después, Sony presentó un nuevo medio de almacenamiento removible: el disquete de 3.5 pulgadas y 720 KB de capacidad DD (Doble Densidad), y posteriormente los últimos disquete de 1.44 MB de capacidad HD (Alta Densidad).

1.4.3. PRIMERA OPCIÓN DE ALTA CAPACIDAD: EL CD-ROM

Los ingenieros de Philips decidieron modificar las especificaciones de los discos compactos de audio para que pudieran almacenar datos de computadora; surge así el formato CD-ROM, **(Compact Disc-Read Only Memory, Disco Compacto de Memoria de Sólo Lectura).** Desde un principio, este formato cautivo a los fabricantes y al público. Un solo disco puede almacenar hasta 640MB de información, a un costo que apenas excede el de un par de disquetes.

1.4.4. EL DVD

Desde un principio, el DVD **(Disc Versatil Digital, disco versátil digital)** fue concebido como un estándar multipropósito; es por ello que se realiza tanto para almacenar información de vídeo (remplazando así el vídeo VHS (Vertical Helical Scan, Exploración Helicoidal Vertical frecuente e incorrectamente llamado Video Home System) como medio de distribución de películas como datos de computadora. Y al contrario de lo que ocurrió con el CD, que originalmente fue utilizado para almacenar audio, y luego se adapto para grabar datos de computadora.

La principal ventaja del disco versátil de comparación con los tradicionales CD-ROM, (Compact Disc-Read Only Memory, Disco Compacto de Memoria de Sólo Lectura) radica en su capacidad de almacenamiento. Mientras que un CD común puede guardar un máximo de 700 MB de datos, un DVD-ROM sencillo puede almacenar hasta 4.7 GB. y gracias a sofisticados trucos para incrementar el volumen de datos de un DVD, los discos más avanzados pueden almacenar hasta 17 GB de datos (más de 24 veces de la capacidad de un CD convencional).

El **DVD Blu-ray** (Disco Versátil de Alta Definición rayo azul), desarrollado principalmente por Sony y Philips, es otro formato de disco óptico de nueva generación de 12 cm de diámetro (igual que el CD y el DVD) para vídeo de alta definición y almacenamiento de datos de alta densidad. De hecho, compite por convertirse en el estándar de medios ópticos sucesor del DVD, su rival es el HD-DVD.

Una capa de disco Blu-ray puede contener alrededor de 25 GB o cerca de 6 horas de video de alta definición más audio, y el disco de doble capa puede contener aproximadamente 50 GB. La velocidad de transferencia de datos es de 36 Mbit/s

El **HD-DVD** (High Definition Digital Versatile Disc, Disco Versátil de Alta Definición) es un formato de almacenamiento óptico desarrollado como un estándar para el DVD de alta definición por las empresas Toshiba, Microsoft y NEC. La capacidad de almacenamiento es de15 GB (unas 4 horas de vídeo de alta definición) y de doble capa, con una capacidad de 30 GB.

1.4.5. TECNOLOGÍA DEL DVD-ROM

Por ser un formato tan avanzado, podría pensarse que su principio de operación es completamente revolucionario. Pero en realidad, un DVD es una evolución del formato CD; al igual que éste, tiene una superficie reflejante en la que se graban minúsculas muescas (pits) en espiral (traces), cuya lectura es realizada por un rayo láser.

La principal diferencia entre un DVD y un CD tradicional, es que en éste se utiliza un rayo láser infrarrojo para realizar la lectura y/o escritura; y en el DVD se recurre a un rayo láser que cae en el espectro visible, en la banda del color rojo.

Así, en el formato del CD la longitud promedio de la muesca es 0.50 micras, mientras que en el DVD se redujo a sólo 0.32micras. Además, se pudo reducir la separación entre pistas adyacentes: de 1.6 a 0.7 micras. Combinando ambos factores y optimizando la codificación de datos almacenados en el disco mediante el formato MPEG (Moving Pictures Experts Group) se logró entonces un disco, DVD, cuyo tamaño es idéntico al de un CD convencional, registre hasta 4.7 GB de datos.

1.4.6. MEMORIA FLASH

La memoria FLASH, mejor conocida como memoria USB (de Universal Serial Bus, en inglés: pendrive o USB flash drive) es un pequeño dispositivo de almacenamiento para guardar la información sin necesidad de baterías (pilas). Esta memoria es resistente a los rasguños y al polvo que han afectado a las formas previas de almacenamiento portátil, como los CD y los disquetes.

Las unidades flash USB fueron inventadas en 1998 por IBM como un reemplazo de las unidades de disquete. Aunque fue un invento de IBM, éste no lo patentó. IBM contrato más tarde a M-Systems para desarrollarlo y fabricarlo en forma no exclusiva. M-Systems mantiene la patente de este dispositivo, como también otras pocas relacionadas.

Las primeras unidades flash fueron fabricadas por M-Systems bajo la marca "Disgo" en tamaños de 8, 16, 32, 64 Y 256 MB. Y en la actualidad se pueden encontrar fácilmente memorias de 1, 2, 4, 8 GB. Estas memorias se han convertido en el sistema de almacenamiento y transporte personal de datos más utilizado, desplazando en este uso a los tradicionales disquetes, y a los CDs.

Esta memoria fue evolucionada de la memoria EEPROM (Electrically Erasable and Programmable Read Only Memory, Memoria de Sólo Lectura Eléctricamente Programable y Borrable), que permite que múltiples posiciones de memoria sean escritas o borradas en una misma operación de programación mediante impulsos eléctricos, frente a las anteriores que sólo permite escribir o borrar una única celda cada vez. Por ello, flash permite funcionar a velocidades muy superiores cuando los sistemas emplean lectura y escritura en diferentes puntos de esta memoria al mismo tiempo.

FIGURA 1.25. MEMORIA FLASH TIPO USB.

A continuación en la figura se presenta un equipo de cómputo el cual esta integrado por:

a) HARDWARE corresponde a componentes físicos y eléctricos de la PC por ejemplo: monitor, mouse, teclado, tarjeta de video, modem, tarjeta madre, microprocesador memoria RAM etc.
b) SOFTWARE corresponde a la parte lógica de la computadora y esta integrado principalmente por tres áreas:
   - Software de sistema (Sistema Operativo, BIOS, Antivirus, etc.).
   - Software de aplicación (Excel, Photoshop, Word, Autocad, etc.).
   - Software específico (generado por programadores para necesidades específicas de clientes, empresas, fábricas u organizaciones).

FIGURA 1.26. COMPUTADORA PERSONAL o PC.

CAPÍTULO 2

LENGUAJES DE PROGRAMACIÓN

2.1. ANTECEDENTES DE LOS SISTEMAS OPERATIVOS

2.1.1. PRIMERA GENERACIÓN (1945-1955): BULBOS Y CONEXIONES

Como vimos en el capítulo anterior después de los infructuosos esfuerzos de Babbage, hubo poco progreso en el diseño de las computadoras digitales, hasta la Segunda Guerra Mundial. A mitad de la década de los cuarentas, Howard Aiken (Harvard), John von Neumann (Instituto de Estudios Avanzados, Pricenton), J. Presper Eckert y William Mauchley (Universidad de Pennsylvania), así como Honrad Zuse (Alemania), entre otros, lograron construir máquinas de cálculo mediante bulbos.

En esos primeros días, uno solo grupo de personas diseñaban, construían, programaban, operaban y daban mantenimiento a cada máquina. Toda la programación se llevaba a acabo en **lenguaje** de **máquina** absoluto, y con frecuencia se utilizaban conexiones para controlar las funciones básicas de la máquina. Los lenguajes de programación eran desconocidos (incluso el lenguaje ensamblador). No se oía de los sistemas operativos, el modo usual de operación consistía en que el programador, reservaba cierto periodo en una hoja de reservación pegada a la pared, iba al cuarto de la máquina, insertaba su conexión en la computadora y pasaba unas horas, esperando que ninguno de los 20,000 o más bulbos, se quemaran durante la ejecución. La inmensa mayoría de los problemas eran cálculos numéricos directos, como ejemplo el cálculo de valores para tablas de senos y cosenos.

A principios de la década de los cincuentas, del siglo pasado, la rutina mejoró un poco con la introducción de las tarjetas perforadas. Fue entonces posible escribir los programas en las tarjetas y leerlas, en vez de insertar conexiones; por lo demás, el proceso era el mismo.

2.1.2. SEGUNDA GENERACIÓN (1955-1965): TRANSISTORES Y SISTEMA DE PROCESAMIENTO POR LOTES

La introducción del transistor a las computadoras a mediados del siglo XX, (aunque fue inventado el transistor en 1948), modificó en forma radical el panorama, ya que las computadoras se volvieron confiables, con la esperanza de que ellas continuaran funcionando lo suficiente, como para realizar un trabajo en forma. Por primera vez, hubo una clara separación entre los diseñadores, constructores, operadores, programadores y personal de mantenimiento.

Para ejecutar un trabajo (es decir, un programa o conjunto de programas), el programador debía primero escribir el programa en hojas de papel, el programa que se utilizó en esta generación fue **EL ENSAMBLADOR** y posteriormente **EL FORTRAN**, después se escribía en formatos específicos y posteriormente se perforaba en tarjetas. Estas tarjetas se debían de llevar en paquete al cuarto de lectura, para verificar su estructura.

Al terminar la computadora el trabajo que estaba en ejecución, un operador pasaba a la impresora, separaba la salida y la pasaba al cuarto de salida, para que el programador la recogiera más tarde. Se desperdiciaba demasiado tiempo de cómputo mientras los operadores caminaban en el cuarto de la máquina.

Dado el alto costo del equipo, no debe sorprender el hecho de que las personas buscaron en forma por demás rápida vías para reducir el tiempo invertido. La solución que, por lo general se adoptó, fue la de **"sistema de procesamiento por lotes"**. La idea subyacente era colectar una charola repleta de trabajos en el cuarto de entrada y leerlas después en una cinta magnética mediante una computadora pequeña, como la IBM 1401, que era muy eficiente en la lectura de cartas, copiado de cintas e impresión de la salida, pero no para los cálculos numéricos. Otra máquina, más cara y específica, como por ejemplo la IBM 7094, se utilizaba para el tiempo oficial.

Después de transcurrir una hora para colectar un lote de trabajos, la cinta se embobinaba y era llevada al cuarto de la computadora, donde se colocaba en una unidad de cinta. El operador cargaba entonces un programa especial (el antecesor del actual sistema operativo), el cual leía el primer trabajo de la cinta y lo ejecutaba. La salida se escribía en una segunda cinta, en vez de imprimirlos. Al concluir cada trabajo, el sistema operativo leía en forma automática el siguiente trabajo y comenzaba a ejecutarlo. Al concluir con todo el lote, el operador retiraba las cintas de entrada y de salida, reemplazaba la cinta de entrada con el siguiente lote y llevaba la cinta de salida a la IBM 1401 para impresión **fuera de la línea** (es decir, sin conectarse a la computadora principal).

Las grandes computadoras de la segunda generación reutilizaron principalmente para los cálculos científicos y de ingeniería; por ejemplo, en la resolución de ecuaciones diferenciales parciales. Por lo general, se programaba en FORTRAN y LENGUAJE ENSAMBLADOR. Los sistemas operativos más comunes eran **FMS** (Fortran Monitor System) e **IBSYS** el sistema operativo de IBM para el equipo 7094.

2.1.3. TERCERA GENERACIÓN (1965-1980) CIRCUITOS INTEGRADOS Y MULTIPROGRAMACIÓN

A principios de la década de los sesenta del siglo anterior, la mayoría de los fabricantes de computadoras tenían dos líneas de productos, distintas y totalmente incompatibles. Por un lado, estaban las computadoras científicas de gran escala, orientadas a palabras, como la ya mencionada 7094, exclusiva para cálculos científicos y de ingeniería. Por el otro, estaban las computadoras comerciales, orientadas a caracteres, como la 1401, de uso común para el ordenamiento de cintas e impresión por los bancos y las compañías aseguradoras.

El desarrollo y mantenimiento de dos líneas de productos completamente distintas, era una propuesta cara para los fabricantes. Además, la mayoría de los nuevos clientes necesitaban una computadora pequeña y accesible en cuanto a costo, pero con el tiempo deseaban un computador más grande y específico, que ejecutaran sus programas anteriores pero de una manera más rápida.

IBM, intentó resolver ambos problemas a la vez; para ello, introdujo el Sistema/360. Este sistema era una serie de máquinas con software compatible, con un rango de la 1401 hasta computadores más poderosos que la 7094. Estos computadores tenían la misma arquitectura y conjunto de instrucciones, al menos en teoría, los programas escritos para una computadora podían ejecutarse en las otras. Además, la 360 se diseñó para hacer cálculos tanto científicos como contables.

La 360 fue la primera línea principal de computadoras que utilizó los circuitos integrados (a pequeña escala), lo que proporcionó una gran variedad de precio y desempeñó con respecto de las computadoras de la segunda generación, construidas a partir de transistores individuales. Tuvo un éxito inmediato; la idea de las computadoras compatibles, y pronto fue adoptado por los demás fabricantes.

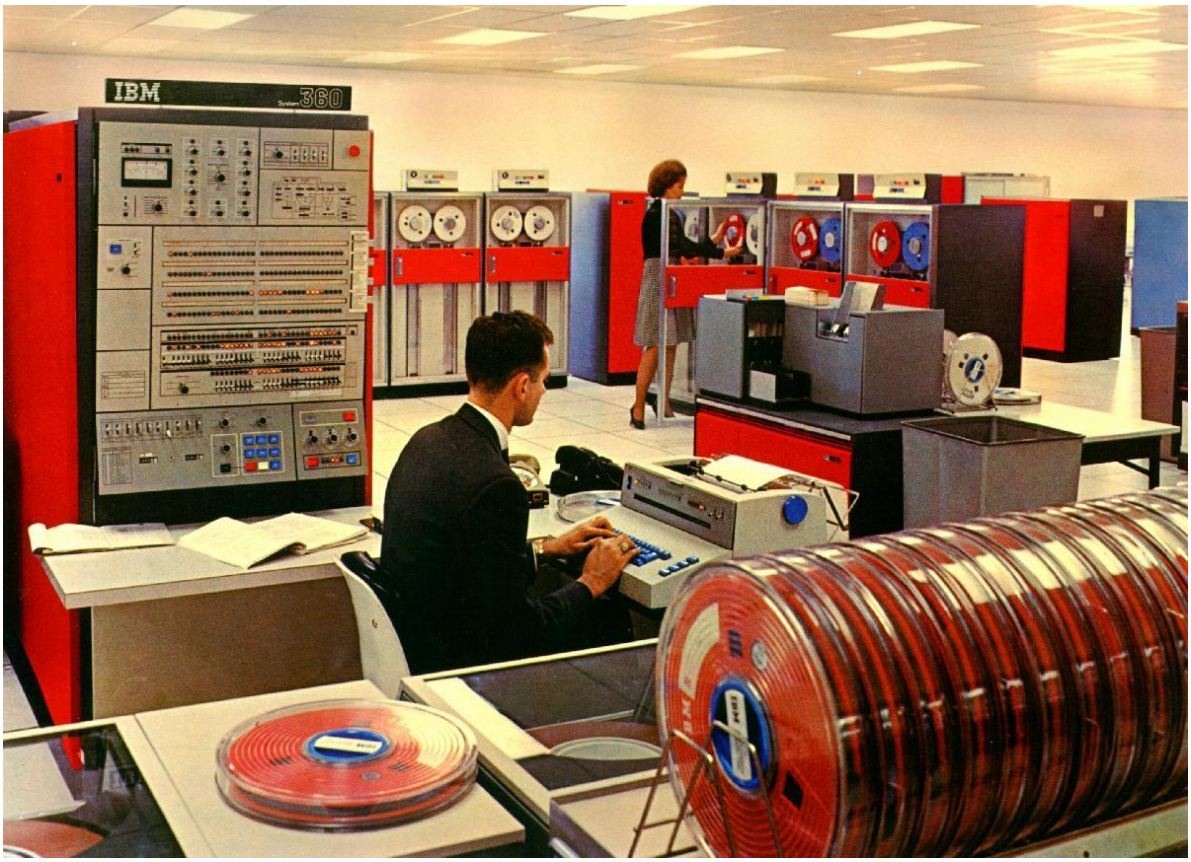


FIGURA 2.1. IBM 360.

La enorme fuerza de la idea de "una familia" era al mismo tiempo su mayor debilidad. La intención era que todo el software (sistema operativo incluido), debía funcionar en todos los modelos. Debía ser compatible para los sistemas con pocos periféricos (modelo 1401) y en sistemas con muchos periféricos. Debía funcionar en los ambientes científicos (modelo 7094) y comerciales. Pero por encima de todo, debía ser eficaz en todos estos diversos usos.

No había forma de que IBM (o cualquiera otra empresa) pudiera diseñar una parte de software que cumpliera con todos estos requisitos en conflicto. El resultado fue un sistema operativo enorme y extraordinariamente complejo, tal vez del doble o triple de magnitud que FMS. Constaba de millones de líneas de lenguaje ensamblador, escrito por miles de programadores, con miles y miles de errores, que requerían de un flujo continuo de nuevas versiones, en un intento por corregirlos. Cada versión actualizada resolvía algunos errores pero introducía otros nuevos.

Uno de los diseñadores de OS/360 (Sistema Operativo/360), Fred Brooks, escribió un sarcástico e incisivo libro (Brooks, 1975), en el cual describe sus experiencias con OS/360, donde la portada del libro muestra a una horda de bestias prehistóricas en un foso con brea.

Sin embargo una de las aportaciones más importante de la tercera generación es la **multiprogramación**. En la 7094, si el trabajo en curso se detenía en espera de una cinta o de que concluyera otra labor de E/S (Entrada/Salida), la CPU solo se detenía a esperar hasta que el proceso entrada/salida (E/S) finalizara.

La solución que se desarrolló fue la de seccionar la memoria RAM en varios bloques, con un trabajo distinto en cada partición, como se muestra en la figura 2.2. Mientras que un trabajo esperaba a que concluyera la E/S, otro podía estar utilizando la CPU. Si se podía mantener en la memoria principal los trabajos suficientes a la vez, la CPU estaría ocupada el 100 por ciento del tiempo.

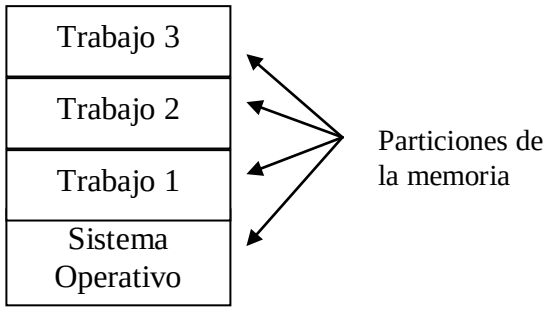


FIGURA 2.2. PARTICIONES DE MEMORIA.

Otra de las características principales de los sistemas operativos de la tercera generación, era la capacidad de leer trabajos de las tarjetas al disco, tan pronto como llegaran a la sala de cómputo. Así, siempre que concluyera un trabajo, el sistema operativo podía cargar un nuevo trabajo del disco en la partición que quedaba desocupada y ejecutarlo. Esta técnica se llama **spooling** (de Simultaneous Peripheral Operation On Line: Operación simultánea y en línea de periféricos) y también se utilizó para las salidas. Con el spooling, las 1401 ya no fueron necesarias y desapareció el traslado de las cintas de un lado a otro.

Sin embargo existía lentitud en el funcionamiento, ya que un solo programador tenia para él solo la máquina a esto se le llama monousuario. Esto provocó el deseo de una rápida respuesta y se preparó el camino para el **tiempo compartido** (timesharing), variante de la multiprogramación, en el que cada usuario tenía una terminal en línea. En un sistema con tiempo compartido, si 20 usuarios están conectados y 17 de ellos están pensando, platicando o tomando café, la CPU puede ser asignada a los tres trabajos que requieren servicio.

El primer sistema serio de tiempo compartido CTSS (Compatible Time-Sharing System Sistema de Tiempo Compartido Compatible), fue desarrollado en MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts).

Después del éxito del sistema CTSS, MIT, Bell Labs y General Electric (que entonces era uno de los principales fabricantes de computadoras) decidieron emprender el desarrollo de una "utilería de computadora", una máquina que soportara a cientos de usuarios con tiempo compartido. Su modelo fue el sistema de distribución de la electricidad (si se necesita la corriente eléctrica, sólo hay que conectarse a un contacto en la pared y, dentro de los límites razonables, con ello se obtiene la corriente de alimentación), es decir, una alta disponibilidad, de manera que el servicio de computación igualara a los servicios de telefonía y a la red eléctrica. Los diseñadores de este sistema, lo llamarón **MULTICS** (MULTiplexed Information and Computing Service).

MULTICS introdujo muchas ideas relevantes sobre la computación, pero su construcción era mucho más ardua de lo esperado. Bell Labs abandonó el proyecto y General Electric se retiró completamente del negocio de la computación. MULTICS llegó a ejecutarse de una manera que fuera útil en un ambiente de producción en MIT y algunos cuantos lugares más, pero el concepto de la utilería de computadora fue un fracaso. Sin embargo aun hoy en día, MULTICS ha tenido una enorme influencia en los sistemas subsecuentes.

Uno de los científicos de Bell Labs que trabajó en el proyecto MULTICS, Ken Thompson, se encontró una pequeña minicomputadora PDP-7 que nadie utilizaba e intentó escribir con la ayuda de Dennis Ritchie, una versión desprotegida de MULTICS para un solo usuario, en **lenguaje ensamblador**, desarrollado en 1969. Este trabajo desembocó en el sistema operativo UNIX (marca registrada de AT&T). Aparte estos mis científicos Ken Thompson y Dennis Ritche crearon el lenguaje de programación C entre 1969 y 1972, como evolución al anterior lenguaje B.

FIGURA 2.3. KEN THOMPSON Y DENNIS RITCHIE

En 1972, se tomó la decisión de diseñar nuevamente UNIX, pero esta vez en el lenguaje de programación C. Este cambio significaba que este lenguaje de programación (UNIX), podría ser fácilmente modificado para funcionar en otras computadoras (de esta manera, se volvía transportable) con lo cual programandose en C podrían desarrollar nuevas adaptaciones y mejorar su sintaxis. Ahora con el lenguaje C, el código era más conciso y compacto, lo que se tradujo en un aumento en la velocidad de desarrollo de UNIX. AT&T puso este sistema operativo, a disposición de universidades y compañías, también el primero en el gobierno de los Estados Unidos de América (EUA), a través de licencias (permisos). En la actualidad domina los mercados de las minicomputadoras y estaciones de trabajo, a nivel mundial.

En Abril de 1999, Ken Thompson Y Dennis Ritchie recibieron “La Medalla Nacional de Tecnología”. La Medalla Nacional de Tecnología es el más alto honor otorgado por el Presidente de los Estados Unidos de América a los principales innovadores. Creada por una ley del Congreso en 1980. La Medalla se otorga anualmente a individuos, equipos, y/o de empresas o divisiones por sus destacadas contribuciones a la nación en los ámbitos económico, ambiental y el bienestar social mediante el desarrollo y la comercialización de los productos de la tecnología, innovación tecnológica, y desarrollo de la nación de la mano de obra tecnológica.

FIGURA 2.4. KEN THOMPSON, DENNIS RITCHIE Y WILLIAM JEFFERSON "BILL" CLINTON

2.1.4 CUARTA GENERACIÓN (1980-1990): COMPUTADORAS PERSONALES.

Con el desarrollo de los circuitos circuitos integrados LCI (Large Scale Integration: Alta Escala de Integración), chips con miles y posteriormente millones de transistores en un centímetro de silicio, se inició la era de la computadora personal. En términos de arquitectura, las computadoras personales no eran muy distintas de las minicomputadoras del tipo de la DPD-11, pero en términos del precio si eran distintas.

Dos sistemas operativos han dominado desde entonces la escena de las computadoras personales y las estaciones de trabajo: **MS-DOS de Micrososft** y **UNIX**.

MS-DOS, tiene un amplio uso en la IBM PC y otras máquinas con el CPU (Unidad Central de Proceso) 8088 de Intel y sus sucesores, 80286, 80386, 80486 y 80586 (mejor conocidos como 286, 386, 486, pentium). Aunque la versión inicial de MS-DOS era relativamente primitiva, las subsecuentes versiones (Windows 3.11, 95, 98, 2000, XP), han incluido características más avanzadas, entre ellas algunas del otro contendiente principal que es UNIX.

UNIX, domina las computadoras que no utiliza Intel, así como las estaciones de trabajo, en particular en las que poseen chips de alto desempeño RISC (Reduced Instruction Set Computer, Computadora con Conjunto de Instrucciones Reducido). Estos computadores tienen en general la potencia de cómputo de una minicomputadora, aunque se dedique a un solo usuario.

Un interesante desarrollo de la cuarta generación, que comenzó a llevarse a cabo a mediados de la década de los ochentas del siglo XX, ha sido el crecimiento de las redes de computadoras personales con **sistemas operativos de red** y **sistemas operativos distribuidos.**

En un **sistema operativo de red**, los usuarios están conscientes de la existencia de varias computadoras y pueden conectarse con equipos de cómputo remotos, y copiar archivos de una máquina a otra. Cada máquina ejecuta su propio sistema operativo local, y tiene su propio usuario (o grupo de usuarios). También los sistemas operativos de red, no tienen diferencias fundamentales con los sistemas operativos de un solo procesador. Es obvio que necesitan un controlador de interfaz de la red, y programas que permitan la conexión y el acceso a un archivo remoto, pero estas características no modifican la estructura esencial del sistema operativo.

Por el contrario, un **sistema operativo distribuido** es aquél que aparece ante sus usuarios como un sistema tradicional de un solo procesador, aun cuando esté compuesto por varios procesadores. En un sistema distribuido verdadero, los usuarios no deben ser conscientes del lugar donde su programa se ejecute o del lugar donde se encuentran sus archivos; eso debe ser manejado en forma automática y eficaz por el sistema operativo. Además esté sistema operativo distribuido requiere de algo más que añadir un poco de código a un sistema operativo de un único procesador, puesto que los sistemas distribuidos y los

centralizados difieren en aspectos críticos. Por ejemplo, los sistemas distribuidos permiten a menudo que un programa se ejecute mediante varios procesadores a la vez, por lo que necesitan algoritmos de asignación del tiempo más complejos, con el fin de optimizar la magnitud de paralelismo.

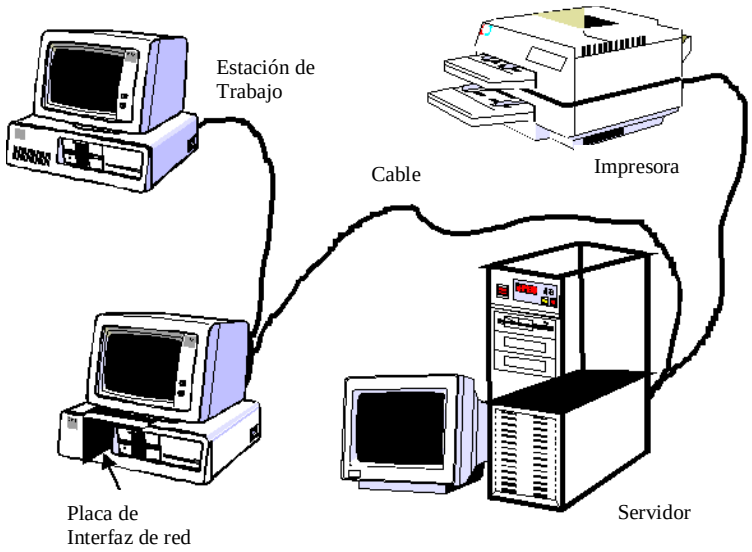


FIGURA 2.5. REDES DE COMPUTADORAS.

2.1.5. QUINTA GENERACIÓN (1990-sin concluir): INTELIGENCIA ARTIFICIAL

La quinta generación de computadoras, fue un proyecto ambicioso lanzado por Japón a finales de los 70´s del anterior siglo. Su objetivo era el desarrollo de una clase de computadoras, que utilizarían técnicas de inteligencia artificial al nivel del lenguaje de máquina, y fueran capaces de resolver problemas complejos, como la traducción automática de una lengua natural a otra (por ejemplo del japonés al inglés).

Sin embargo por no haber alcanzado, muchos de sus objetivos propuestos para esta generación, este campo sufrió una interrupción en la última década anterior. En la actualidad se sigue estando tan lejos de cumplir la prueba de Turing como cuando se formuló: "Existirá Inteligencia Artificial cuando no seamos capaces de distinguir entre un ser humano y un programa de computadora en una conversación a ciegas". A pesar de no llegar a todos sus objetivos la quinta generación de computadoras, se dieron avances muy importantes en las áreas de la Inteligencia Artificial como: los **sistemas expertos, robótica, reconocimiento de patrones, redes neuronales, realidad virtual.**

FIGURA 2.6. ALAN TURING.

Como anécdota, muchos de los investigadores sobre IA sostienen que "la inteligencia es un programa capaz de ser ejecutado independientemente de la máquina que lo ejecute, computador o cerebro".

2.2. UN PROGRAMA DE COMPUTADORA

Cuando se escribe un programa se le observa desde dentro. Se conoce cómo y porqué trabaja de tal manera. Es una forma diferente y más excitante de ver un programa de computadora. No importa cuanta experiencia se tenga en la ejecución de programas, ni cuan documentado se esté, escribir un programa presenta retos nuevos por completo.

Un programa no es otra cosa que una serie de instrucciones para su computadora, indicándole qué hacer y precisando el orden correcto para hacerlo, en un leguaje que la computadora es capaz de entender.

2.2.1. LENGUAJES DE PROGRAMACIÓN

2.2.1.1. Lenguajes de bajo nivel

Los lenguajes de bajo nivel son lenguajes de programación que se acercan al funcionamiento de una computadora. El lenguaje de más bajo nivel es, por excelencia, el **código máquina**. A éste le sigue el **lenguaje ensamblador**, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.

El lenguaje de máquina, es el sistema de códigos directamente interpretable por un circuito microprogramable, como el microprocesador de una computadora o el microcontrolador de un autómata (un PLC).

Estos códigos lo utilizan los circuitos microprogramables son sistemas digitales, lo que significa que trabajan con dos únicos niveles de tensión. Dichos niveles, por abstracción, se simbolizan con el cero, 0, y el uno, 1, por eso el lenguaje de máquina sólo utiliza dichos signos. Esto permite el empleo de las teorías del álgebra booleana y del sistema binario en el diseño de este tipo de circuitos y en su programación.

Cuando un programa le indica a la computadora que despliegue un mensaje en la pantalla o lo oprima, el microprocesador envía las señales electrónicas adecuadas. Estas señales le indican en que parte de la memoria encontrar el mensaje y adónde enviarlo.

Técnicamente, el microprocesador solo puede realizar cuatro acciones. Puede **mover** datos desde una posición de memoria hasta otra, **cambiar** los datos almacenados desde una ubicación de memoria, **determinar** si la memoria contiene un dato específico y **alterar** la secuencia de las instrucciones que está ejecutando. Todas estas acciones se realizan enviando, recibiendo o supervisando pulsos electrónicos.

La computadora trabaja con estos pulsos en uno de dos estados (niveles de voltaje); una señal electrónica podrá estar encendida o apagada. Para ejecutar una tarea alimenta varias señales en estado encendido o apagado al microprocesador.

A fin de escribir instrucciones en este nivel básico, se utiliza el número 0 para representar el estado apagado y el 1 para encendido. A estos valores se les denomina dígitos binarios (bits) o instrucciones binarias, basadas en el sistema de numeración binario que utiliza varias combinaciones de 0 y 1 para representar todos los valores.

Cuando nacieron las computadoras y, por consiguiente, la necesidad de controlarlas, la programación se realizaba por el manejo directo (encendido y apagado) de estas señales.

Con la evolución de las computadoras fue posible alimentar la totalidad de un programa en la computadora y que está ejecutara las instrucciones correspondientes. Se requerían miles de combinaciones de 1s y 0s, hasta que se desarrolló el leguaje ensamblador.

El lenguaje ensamblador utiliza códigos mnemónicos para representar tareas en la computadora que están relacionadas de manera directa con el equipo. Un código mnemónicos es una palabra o abreviatura fácil de recordar, que representa una tarea completa de microprocesador. Por ejemplo, el código MOV le indica a la computadora mover datos de una ubicación de memoria a otra. El código JMP le indica que salte a otra posición en la memoria (memoria baja y memoria alta de la RAM). De esta manera, en lugar de escribir una serie de 0s y 1s, el programador de lenguaje ensamblador puede utilizar códigos como MOV y JMP.

2.2.1.2. Lenguajes de medio nivel

Existen además lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.

2.2.1.3. Lenguajes de alto nivel

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés.

Actualmente, la mayoría de los programadores utilizan un leguaje de alto nivel, cuyas instrucciones son palabras que la gente entiende, y no simples mnemónicos o combinaciones de 1s y 0s. Cada palabra representa de manera completa una operación práctica y no una tarea del microprocesador. Por ejemplo, en C las funciones puts (para colocar una cadena de caracteres en la pantalla) que indica a la computadora que despliegue cierta información.

La misma función requeriría un buen número de nemónicos en lenguaje ensamblador y, quizá, cientos de pulsos electrónicos binarios ó bits a utilizar.

Ejemplo de instrucciones comparables en C, lenguaje ensamblador y código binario.

En C:

Puts (“TESIS”)

En ensamblador:

```
        MOV   AH,   9
        MOV   DX, OFFSET NAME
        INT     21h
        MOV    AG,   4Ch
        INT     21h
NAME  DB    “TESIS”
```

En binario o lenguaje máquina:

```
11001001010000000000000100011011110100000000000010000000001100010001 1
10110010100000111100101000010011100000001000111111110001010100001100 0
01011111000101000010101111111000010101011100010100111010010000101 0
```

En el ejemplo anterior muestra una sencilla instrucción en C/C++ para desplegar una palabra en la pantalla, y sus equivalentes aproximados en Lenguaje Ensamblador y en código binario.

Por supuesto, la computadora realmente no entiende lo que significa PUTS o cualquier otro comando en un lenguaje de alto nivel. Antes de que la computadora pueda ejecutar el comando, esté deberá ser traducido al lenguaje propio de la computadora, el binario.

Esta traducción de palabras de uso común por el ser humano a instrucciones binarias se realizan de dos maneras: **compilando** ó **interpretando**.

2.2.2. COMPILADORES

Al compilar se traduce un programa completo a la vez, y se graba el resultado en el disco, de tal manera que se pueda ejecutar posteriormente el programa.

Supongamos que se escribe un informe en inglés para presentarlo a un grupo de estudiantes de la UNAM. Se contrata a un traductor para que reescriba el informe en español, se hacen tantas copias como se requieran y posteriormente se distribuyen. Si el traductor encuentra errores de tipo gramatical, los identificará y deberán ser corregidos a fin de que se pueda continuar con la traducción del informe. Ahora bien, si dentro de un año se vuelve a requerir el mismo informe, todo lo que se requiere hacer es copiar y distribuir la traducción completa. No se requiere de nueva cuenta al traductor.

El compilador hace lo mismo con un lenguaje de computadora. Es un programa que (con la ayuda de otro programa llamado encadenador) convierte todas las instrucciones de un programa a código binario, de tal manera que pueda ejecutarse. El compilador se asegura que el programa se apegue a las reglas del lenguaje C o C++, y crea un archivo objeto, que es un forma intermedio del programa. Durante el proceso, el compilador indicará si se encuentra una instrucción que no entiende. En este caso se deberá corregir el problema y rehacer la compilación. El encadenador convertirá el código objeto en un programa ejecutable. Este proceso no ejecuta el programa, solo realiza la traducción.

Cuando se utiliza un compilador, el programa existe en tres estados. Las introducciones del lenguaje C se almacena en un archivo de texto llamado **Código Fuente**, este archivo pude ser leído e impreso, desde la misma manera que un documento creado por un procesador de textos. Se puede modificar este archivo para hacer cambios al programa. El programa

compilado estará contenido en un **archivo objeto** y el resultado final será un **archivo ejecutable**, que puede correrse o ejecutarse cuantas veces se requiera.

C, C++, VISUAL, PASCAL, COBOL, ENSAMBLADOR, PROLOG Y FORTRAN son ejemplos de lenguajes compilables.

2.2.3. INTÉRPRETES

Un intérprete traduce cada instrucción antes de ejecutarla. Consideremos el caso del informe en inglés.

En este caso se escribe el informe en ingles y se contrata a un traductor. Mientras el público escucha, el intérprete traduce la primera frase, en español y la lee al auditorio.

Después, el intérprete traduce y lee la segunda frase, y así sucesivamente, hasta finalizar la lectura del informe. Hay que considerar que en este caso, el informe sólo existe en inglés. Si se requiere volver a presentar el informe dentro de un año, se tendrá que recontratar a un intérprete y repetir lo realizado.

Con un intérprete, el programa no existe más que en su versión en el código fuente original.

Algunos ejemplos de intérpretes son: **BASIC, Perl, PHP, Logo, Python**.

2.2.4. EL COMPILADOR C/C++

C es un lenguaje compatible. Es decir es una colección de comandos y funciones, en forma de palabras de aspecto familiar, que se convierten en instrucciones binarias que pueden ejecutarse por la computadora.

2.2.5. VELOCIDAD

C es un lenguaje de alto nivel “mas cercano” al ensamblador ya que algunos comandos de C direccional de manera directa al equipo físico de la computadora (por ejemplo en lo puertos directamente en la computadora como lo veremos más adelante). Esto también da lugar a que los programas compilados en C se ejecuten muy rápidamente. De hecho, su ejecución es tan rápida que se puede utilizar C para escribir sistemas operativos, aplicaciones relacionadas con comunicaciones e ingeniería e, incluso a compiladores.

2.2.6. TRANSPORTABILIDAD

También se pueden codificar programas muy rápidos por medio del lenguaje ensamblador, sin embargo, los mnemónicos varían con cada familia o versión del microprocesador.

C utiliza un conjunto de comandos estandarizados. En casi todos lo casos, el programa se escribe una sola vez sin importar la plataforma (computadora o sistema operativo) en el que se requiere ejecutar. Mientras que el código fuente no cambia, y lo único que se requerirá será un compilador para ese tipo de plataforma.

2.2.7. BIBLIOTECAS DE FUNCIONES

La potencia de C viene de las bibliotecas de funciones proporcionadas con el compilador. Una **función** es un conjunto de instrucciones que realizan una tarea específica. Una **biblioteca** es un archivo individual, proporcionado con el compilador, que contiene funciones para tareas comunes, de tal manera que no es necesario escribirlas.

Por ejemplo, C no tiene un comando para desplegar información en la pantalla. Esta, sin embargo, es una tarea tan común que varias funciones para desplegar información se proporcionan con la biblioteca de C. Así en lugar de tener que escribir la función, se utiliza una de las proporcionadas con el compilador.

CAPÍTULO 3

LENGUAJE C

3.1. CONCEPTOS BÁSICOS DE "C, C++"

3.3.1. ESTRUCTURA DE UN PROGRAMA C

Un programa C contiene una o más funciones. Cabe recordar que una función es un conjunto de instrucciones dadas a la computadora para realizar una tarea específica. Muchas de la funciones que se realizan ya están escritas, compiladas e incluidas en la biblioteca de funciones proporcionada con el compilador. En lugar de escribir las instrucciones de manera individual, se instruye al compilador para que utilice una de las funciones estándar. Sólo cuando se requiera ejecutar una tarea que no existe en la biblioteca es necesario escribir una función particular.

Todos los programas C y C++ deberán comenzar con la función llamada main ( ).

Las instrucciones pueden incluir comandos C, los nombres de las funciones en la biblioteca o aquellas escritas por el programador. Una llave o corchete que abre ({) deberá aparecer antes de la primera instrucción, y una que cierra (}) deberá seguir la ultima instrucción.

Las llaves que abren y cierran ({ }) se denominan delimitadores e indican el principio y el final de un bloque, o sección de código.

El siguiente será un programa C/C++ que escribe la palabra TESIS en la pantalla:

```
main ( )
{
        Pust("TESIS");
}
```

Este programa contiene una sola instrucción pero obstante debe seguir las reglas establecidas en C. Las comillas que encierran "TESIS" no se desplegarán, sólo indican que se desplieguen los caracteres contenidos entre ellas, y no una constante o variable cuyo nombre es "TESIS".

El punto y coma después de la instrucción se denomina **terminador del enunciado**. Dicho punto y coma le indica al compilador que ha llegado al final de una instrucción.

Esto significa que cada línea deberá terminar con un punto y coma. En algunas ocasiones las instrucciones ocupan más de una línea en la pantalla. En lugar de colocar el punto y coma de la línea en la pantalla, se incluirá al final de la instrucción.

C y C++ se denominan lenguajes de formato libre. Esto significa que no se toma en consideración la ubicación de las llaves o el inicio de las líneas de instrucción. Por lo tanto, es factible escribir el programa como:

main ( ) {puts(“TESIS”);}

3.1.2. MAYÚSCULAS, MINÚSCULAS

Otra convención de programas C es el título para el uso de las mayúsculas. Los comandos y nombres de función siempre se escriben en minúsculas.

El uso de mayúsculas está reservado para elementos tales como *constantes y nombres simbólicos.*

3.1.3. LA FUNCIÓN return( )

¿Qué sucede una vez que la computadora termina de ejecutar las instrucciones de un programa? El programa de detiene y la computadora retorna al estado que estaba antes de ejecutarlo. Si el programa se ejecutó desde el llamador del sistema operativo, éste reaparecerá. Si se ejecutó desde Windows, reaparecerá la cubierta del escritorio.

El regreso del control al sistema operativo ó a Windows ocurre de manera automática. Algunos compiladores, sin embargo, requieren que los pasos a ejecutar se definan de manera explícita, incluyendo el regreso al sistema. Para estos compiladores, se deberá incluir el comando return(0) o return 0, justo antes de la llave que cierra correspondiente a la función main( ):

```
main( )
{
        puts(“Estoy Bien”)
        puts(“Tu no lo estás”);
        return(0);
}
```

La función return(0), le indica a la computadora que regrese al sistema anfitrión.

3.1.4. LOS PARÁMETROS

Un parámetro es un elemento de información que es requerido por una función para hacer un trabajo. Por ejemplo, puts( ) es una función que existe en la biblioteca. La función

contiene instrucciones que le indican a la computadora que despliegue una cadena de caracteres en la pantalla. Pero, ¿qué cadena de caracteres despliega? Se le tiene que indicar lo que se desea desplegar colocándolo dentro de los paréntesis. A esto se le denomina pasar un parámetro. Por ejemplo la siguiente instrucción.

puts(“Tesis”);

3.1.5. COMANDO #include

El comando #include es una directiva, que le indica al compilador que utilice la información contenida en el archivo *stdio.h*, al que se considera como un archivo encabezado. El nombre *stdio* es una abreviatura de standard input/output (entrada/salida estándar), y el archivo stdio.h contiene todas las instrucciones que requiere el compilador para trabajar con archivos en discos o externos y para poder mandar información a la impresora. La instrucción para utilizar un archivo de encabezado deberá ser dada antes de main( ).

Por ejemplo, para utilizar la función getchar( ) en un programa, se debe utilizar el comando #include para stdio.h. Además se deberá incluir en todos los programas para evitar errores.

3.2. VARIABLES Y CONSTANTES

Los diferentes tipos de datos utilizan, mayor o menor espacio en memoria. Por otro lado, no todas las funciones de C pueden ejecutarse con todos los tipos de datos. Se obtienen errores de compilación o de ejecución, cuando a un programa se le intenta contestar con una palabra y esta esperando un número, de ahí su importancia de establecer exactamente que tipo de dato va a utilizar el programa.

3.2.1. DATOS CARACTER

Los valores de datos de tipo carácter – abreviado char – pueden ser letras individuales, números u otros caracteres de teclado. Para cada elemento de datos tipo char la computadora reserva espacio suficiente para almacenar un carácter individual. Por lo tanto, si se requiriera cinco diferentes elementos de datos char, el computador reservará cinco espacios de memoria por los caracteres.

FIGURA 2.7. POSICIONES DE MEMORIA.

Sin embargo, un elemento de datos char también puede ser un dígito numérico, tal como el 1,2 ó 3. Pero hay una distinción entre el carácter "1" y el número 1. Como carácter "1" no puede ser utilizado en una operación matemática por que su valor no es numérico, y como número 1 puede ser utilizado en las operaciones matemáticas básicas (* , / , +, -)

3.2.2. CADENAS

Una cadena es un grupo de caracteres, tales como una palabra, frase o enunciado. C no tiene definido un tipo de datos para las cadenas, como otros lenguajes. En su lugar se trabaja con una cadena como un conjunto de valores de tipo char, utilizando lo que se conoce como un arreglo.

3.2.3. ENTEROS

Un número entero – abreviado int – es un número sin decimales. Podrá ser un número positivo, negativo o cero, pero no podrá tener posiciones decimales.

Un axioma común de C es "Use un int para contar." Utilice un int para contar el número de casos de algo o para contar cualquier cosa que solo exista en unidades enteras.

Cada elemento de datos int requiere el espacio de almacenamiento de dos caracteres, no importa si el valor del número es 2 ó 2000. Pero a fin de utilizar sólo dos espacios para cada número, C debe limitar el valor de los datos int desde – 32,768 hasta 32,767. Los números fuera de este rango requieren más de dos espacios para su almacenamiento.

short int	Un número entero positivo entre 0 y 255
int	Un número entero entre – 32,768 y 32,167
long int	Un número entero entre – 2,147,483,648 y 2,147,483,647
unsigned log	Un número entero positivo entre 0 y 4,294,967,295

3.2.4. NÚMEROS DECIMALES

Aquellos números que pueden tener decimales se llaman valores de punto flotante, por lo que C tiene un tipo de datos float o flotante para trabajar con estos números decimales. Debido a que los números de punto flotante pueden ser demasiado pequeños o grandes, sus rangos generalmente se expresan con números exponenciales. Por ejemplo, un valor float puede ser un número tan grande como 3.4E+38. Esta notación indica, "Desplace el punto decimal 38 lugares a la derecha, agregando tantos ceros como sea necesario".

Existen tipos de datos adicionales para manejar valores aún mayores de números:

float	números desde 3.4E – 38 hasta 3.4E+38
double	números entre 1.7E – 308 y 1.7E+308
long double	números entre 3.4E – 493 y 1.1E+4932

3.2.5. DECLARACIÓN DE UNA CONSTANTE

Declarar una constante significa indicar al compilador C nombre de la constante y su valor. Esto se hace antes de la función main( ), utilizando la directiva del compilador #define, cuya sintaxis general es:

```
#define    Nombre    Valor
```

No se utiliza punto y coma después de la directiva, y deberá existir al menos un espacio entre la directiva, el nombre de la constante y el valor asignado.

3.2.5.1. ¿Por qué Constantes?

Si el valor de un elemento no va a cambiar en un programa, ¿Por qué preocuparse utilizando una constante? ¿Por qué no utilizar en forma directa el valor de una instrucción de programa? Por ejemplo, si el principio del programa se establece:

```
#define    TELEFONO    "555-1234"
```

Se podrá desplegar el valor del número de teléfono utilizando un comando como

```
puts    (TELEFONO);
```

Pero, ya que el valor del número telefónico no se modificará dentro del programa, también se puede desplegar de manera directa:

```
puts    ("555.1234");
```

Después de todo, el resultado será exactamente el mismo y evita tener que escribir la directiva #define para establecer el nombre y el valor de la constante.

Sin embargo, utilizando constantes el programa es más fácil de modificar. Por ejemplo, suponiendo que el programa tenga 20 veces la instrucción para desplegar el número de teléfono. Si es necesario cambiar dicho número y no se ha utilizado una constante, será necesario encontrar y modificar los comandos puts( ). Puede suceder que por error sólo se modifiquen 19 de los números, dando lugar a que en algún caso de valor desplegado sea incorrecto.

3.2.6. DECLARACIÓN DE UNA VARIABLE

Declaración de una variable significa proporcionarle al compilador C el nombre y el tipo de variable, y de manera explícita se deberá especificar el tipo de dato para el valor de la variable. La sintaxis general para declarar una variable es:

tipo nombre;

El número de espacios entre el tipo y el nombre es a criterio del programador, siempre y cuando al menos deje uno.

Se separan los nombres de las variables con comas y se terminan la declaración con un punto y coma, como en:

```
main ( )
{
        int contador, hijos, fecha;
        float  tasa_impuesto, descuento;
}
```

Las variables se declaran en un grupo dentro de la función main ( ), después de su correspondiente llave que abre y antes que otras instrucciones.

3.2.6.1. Asignación de Valores

Algunas variables tienen un valor inicial – el valor que se requiere tenga la variable al inicio del programa. A diferencia de una constante, este valor puede cambiar mientras el programa se ejecuta. El valor inicial se asigna ya sea en la declaración o en una instrucción separada.

Se puede asignar el valor como parte de la declaración:

```
main ( )
{
      int  contador=5;
```

```
chair  inicial=`A´;
float  tasa=0.55;
```

3.2.6.2. Declaración de Cadenas Variables

C permite el manejo de datos tipo cadena por medio de arreglos.

Una cadena es un conjunto de caracteres. Eso es exactamente lo que es una cadena – un conjunto de variables char agrupadas en algo denominado un arreglo. Los caracteres que conforman la cadena se almacenan juntos, en posiciones consecutivas de memoria. Una variable que tiene valor "TESIS" se verá tal como se muestra en la figura.

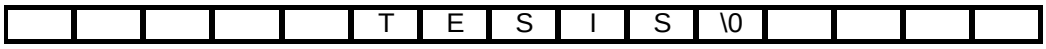


FIGURA 2.8. POSICIONES DE MEMORIA.

El símbolo \0 es un código especial que C inserta después de una cadena. Este carácter marca el final de la cadena, de manera que funciones tales como puts ( ) saben hasta qué punto siguen extrayendo caracteres para desplegarlos.

Para declarar una variable tipo cadena se utiliza el tipo de variable char y se especifica el número máximo de caracteres que podrá contener la variable. La sintaxis es la siguiente:

```
Char   NOMBREVAR[N];
```

Donde el nombre de la variable es NOMBREVAR y N es el número máximo de caracteres que contendrá. El número siempre se especificará entre paréntesis cuadrados.

El número especificado deberá será realmente uno mayor que el máximo de caracteres que se desee utilizar. C requiere el espacio adicional para almacenar el código/0.

Se podría declarar la variable con un valor más grande, tal como:

```
 char      cliente[80]
```

Pero si se tienen varias variables como está estaría desperdiciando un poco los recursos de la computadora.

Una vez declarada una variable cadena se le puede asignar valor. Como con todas las variables. Se le puede asignar un valor inicial, aceptarlo desde el teclado o de un archivo en externo.

3.3. SALIDA EN C/C++

En general, cuando hablamos de salida nos referimos al copiado de datos en la memoria de la computadora a otra ubicación, desplegándolos en la pantalla, imprimiéndolos o grabándolos en un archivo en disco.

3.3.1. FUNCIÓN puts( )

La función puts ( )es la que se encarga de desplegar una cadena en el monitor. El parámetro (la información entre paréntesis, que se desea desplegar) deberá ser una de las siguientes variantes para datos tipo cadena:

- Una literal cadena:

```
puts("Tesis");
```

- Una constante cadena:

```
#define MENSAJE  "Tesis"
main( )
{
puts(MENSAJE);
}
```

- Una variable cadena:

```
char  saludo[ ]="¿Como estas?"
main( )
{
 puts(saludo);
}
```

3.3.2. FUNCIÓN putchar( )

La función putchar( ) despliega sólo un valor tipo char en la pantalla. El parámetro deberá ser uno de los siguientes, considerando que el tipo del dato deberá ser char:

- Una literal:

```
putchar("H");
```

- Una constante:

```
#define INICIAL “H”;
main( )
{
putchar(INICIAL);
}
```

- Una variable:

```
main( )
{
char letra;
 letra=(letra);
}
```

Solo se permite un carácter. La instrucción

```
putchar(‘Si’);
```

dará lugar a un error de compilación.

3.3.2.1. Código Nueva-Línea

La secuencia \n ejecuta un comando de nueva-línea, con lo que avanzará el cursor al principio de la siguiente línea. Se emplea este comando para mover el cursor a la siguiente línea después de que putchar( ) despliega un carácter. En el caso de las instrucciones:

```
putchar(‘A’);
putchar(‘\n’);
```

se desplegará en la pantalla la letra A y el cursor se desplegará al principio de la siguiente línea. Obsérvese en el código \n, debe ir entre apóstrofos.

Mientras que en el código \n con frecuencia se utiliza al final de la instrucción, puede aparecer en cualquier posición dentro de las comillas del parámetro.

El comando

```
puts(“A\nB\nC”)
```

desplegará tres líneas de texto:

A
B
C

3.3.3. FUNCIÓN printf( )

Las funciones puts( ) y putchar( ) son muy útiles pero tienen severas limitaciones. Ninguna de las dos puede desplegar datos numéricos y ambas pueden desplegar tan sólo un argumento, o parámetro. Tanto puts( ) como putchar( ) sólo puede desplegar una cosa.

C y C++ poseen una función más versátil llamada printf( ). Esta función puede desplegar datos de cualquier tipo y puede trabajar con múltiples argumentos en un parámetro. Aunado a esto printf( ) puede controlar cómo aparecerán los datos (dar formato a la salida).

En un nivel básico, puede usarse printf( ) en lugar de puts( ) para desplegar una cadena.

3.3.3.1. Desplegado de Números

A fin de comunicar datos numéricos y para dar formato a todos los tipos de datos, se deberá dividir el parámetro de la función printf( ) en las dos partes mostradas.

La cadena de control, da formato entre comillas, indica al compilador donde se desea que aparezcan los datos en la línea desplegada. Incluye cualquier literal de texto que se desee desplegar, así como marcadores de espacio, denominados especificadotes de formato, que indican el tipo y la posición de los datos.

Cada especificador empieza con el símbolo por ciento (%) seguido por una letra que identifica el tipo de dato:

%d Para desplegar un número entero
%u Para desplegar un entero sin signo
%f despliega un número decimal, float o double
%e despliega un número en notación científica
%g despliega un número en la más corta de sus dos representaciones, decimal o científica
%c despliega un valor char
%s despliega una cadena

Por lo tanto, la primera parte de una instrucción printf( ) podrá verse como sigue:

printf(“%d”);

El símbolo % le indica al compilador que a continuación encontrará un especificador de formato.

La segunda parte del parámetro es la lista de datos, la que contendrá los valores, constantes o variables que se desea desplegar. El separador entre la lista de datos y la cadena de control es una coma, y recoloca una coma entre cada elemento de datos. Cuando el compilador crea el código objeto sustituye los datos en la lista por los separadores de espacio.

Una instrucción printf( ) completa, pero muy sencilla, se verá como esta:

```
printf("%d",12);
```

Cuando dicha instrucción se ejecuta el valor 12 se insertará en le posición del separador de espacio, que es el especificador de formato %d.

La cadena de control puede también incluir texto junto con los dos especificadotes de formato para desplegar datos. Como ejemplo consideremos está línea:

```
printf("Tengo %d años de edad",12);
```

Aquí, la cadena de control es:

```
"Tengo %d años de edad"
```

Pero, para combinar texto con una constante o literal numérica se requiere utilizar printf( ) y un especificador de formato, como en el programa siguiente:

```
  main( )
{
        int edad;
        edad = 12;
        printf("Tengo %d años de edad", edad);
}
```

Se puede asignar cualquier número de parámetros para desplegar múltiples argumentos, pero se deberá incluir un especificador de formato para cada argumento. Los valores en la lista de datos deberán aparecer en el mismo orden que en sus especificadores. El primer elemento en la lista sustituirá al primer especificador de formato, el segundo al siguiente y así sucesivamente. Véase en el siguiente programa:

```
main( )
{
  int   suerte_1, suerte_2;
  suerte_1 = 12;
  suerte_2 = 21;
  printf("Mis números de la suerte son %d y %d", suerte_1, suerte_2;
}
```

Y despliega la siguiente información en la pantalla:

Mis números de la suerte son 12 y 21

3.3.3.2. Para Formatear la Salida

Controlarse la separación y el número de caracteres desplegados utilizando especificadores de ancho de campo. Por ejemplo, sin un especificador de ancho de campo los números tipo float se despliegan con seis decimales. Esta es la razón por la que la instrucción:

```
printf("El costo es de %f  por %d piezas", importe, piezas);
```

la siguiente instrucción despliega en pantalla

El costo es de 45.580000 por 5 piezas

Para especificar el número de decimales se emplea el formato, es decir n=al número de decimales que desplegara la pantalla.

```
%.nf
```

donde *n* es el número de decimales, deseado. Por ejemplo,

```
printf("El costo es de %.2f", importe);
```

desplegará el importe cuyo tipo float, con solo dos posiciones decimales:

```
El costo es de 45.58
```

3.4. ENTRADA EN C/C++

Entrada es el proceso de ingresar datos a la computadora para su utilización por el programa. Cuando se ingresan datos su valor se almacena en una variable. Por ejemplo, los datos que un usuario ingresa como respuesta a un mensaje con solicitud de respuesta

"llamador" vienen a ser el valor de una variable que se almacena en memoria. Posteriormente, se utiliza la variable en cualquier proceso que realice el programa. En la entrada puede provenir de diversas fuentes.

La entrada adecuada de datos es crítica para un programa. Como dice el antiguo adagio: basura en la entrada, basura a la salida. Si los datos ingresados de un programa son erróneos la salida será incorrecta. La exactitud de la salida no podrá ser mejor que la de la entrada.

Sólo se puede ingresar datos de una variable, nunca en una constante. La constante siempre contendrá el valor que el fue asignado. Cuando se ingresa el valor en una variable, realmente se está colocando el valor en un área de memoria que está asociada a dicha variable.

3.4.1. LA FUNCIÓN gets( )

La función gets( ) ingresa una cadena en una variable. El parámetro es el nombre de la variable. Por ejemplo:

```
main( )
{
        char nombre[15] ;
        gets(nombre);
        puts(nombre);
 }
```

Cuando se ejecuta una instrucción gets( ), el programa aparentemente se detendrá. En realidad estará esperando que el usuario teclee algo. En realidad, nada de lo tecleado se convertirá en el valor de la variable hasta que se presione Enter(↵).

El carácter ↵ no se asigna a la variable, pero C agrega el terminador \0 para completar la cadena.

3.4.2. FUNCIÓN getchar( ) Ó getche EN BORLAN C

La función getchar( ) ingresa un solo carácter desde el teclado. Con la mayor parte de los compiladores se pueden ingresar caracteres ya sea en variables tipo int o char. Para ingresar un carácter se puede emplear cualquiera de estos formatos:

```
int letra                    char letra
letra = getchar( );          letra = getchar( );
```

Cuando el usuario oprime una tecla getchar( ) se captura en la pantalla. No se requiere a continuación un ↵ , ya que getchar( ) sólo acepta un carácter el programa continuará inmediatamente después de que uno haya sido ingresado.

3.4.3. OPERADOR DE DIRECCIÓN &

Se ha mencionado que cada variable utilizada en un programa ocupa espacio en la memoria de la computadora. (La cantidad de espacio depende del tipo de la variable). Cada posición o espacio de memoria está numerado, a partir de 0 y hasta la última posición del sistema. Si una computadora tiene 2 megabytes de memoria, las posiciones estarán numeradas desde 0 hasta 2,097,151 tal como se muestra en la figura 2.9. A estos números se le conoce como direcciones de memoria.

Cuando se declara una variable, C reserva suficientes posiciones de memoria para almacenar el valor. Si, por ejemplo, se declara una variable tipo int a la que se da el nombre de contador, C reserva dos posiciones de memoria para almacenar el valor de contador. La primera de estas dos posiciones se conoce como la dirección de la variable.

Como se muestra a continuación el orden secuencial de las direcciones de la memoria principal, es decir en la memoria RAM.

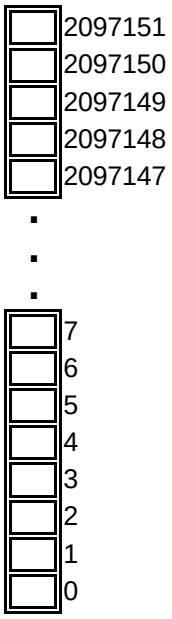


FIGURA 2.9. DIRECCIONES DE MEMORIA.

En la figura 2.10 la variable contador tiene el valor 2341. Este valor se almacena en las posiciones 21560 y 21561 de la memoria. Por lo tanto, la dirección del contador, no es su valor (2341), sino 21560.

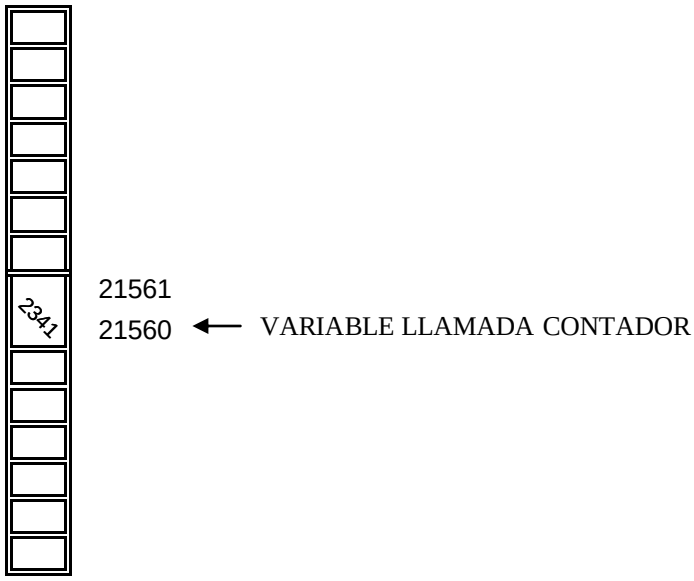


FIGURA 2.10. UNA VARIABLE Y SU DIRECIÓN.

Cuando se hace referencia al valor de una variable, quizá para desplegarla en un comando de salida, se utiliza el nombre de la variable. También se puede hacer referencia a la dirección de la variable precediendo con un ampersand (&) el nombre de la variable, como en &contador. El carácter & se denomina el operador de dirección. Por medio de él se indica a C que está interesado en conocer la dirección donde esta almacenada la variable, y no el valor almacenado en dicha dirección. De acuerdo con esto y considerando el mismo ejemplo, contador tiene el valor 2341 y el valor de &contador es de 21560.

No se utiliza el operador de dirección con variables de tipo cadena. Las cadenas son una clase especial de variable llamada arreglo, lo que se verá con mayor detalle más adelante.

3.4.4. FUNCIÓN scanf( )

La función scanf( ) es un método de propósito general para ingresar información de todos los tipos.

Esta función escudriña (supervisa) las pulsaciones que se ingresan por el teclado, posteriormente interpreta lo obtenido basándose en los especificadores de formato. Como printf( ), scanf( ) puede manejar múltiples argumentos, de manera que se pueden ingresar variables de tipo numérico, char o cadena al mismo tiempo.

En la función scanf( ) a los especificadores se les llama caracteres de conversión. Los especificadores de formato son los mismos que se utilizan con printf( ):

%d estrada de un número entero
%u entrada de un entero sin signo
%f entrada de un número float
%e entrada de un número en notación científica
%g entrada de un número decimal en su presentación más corta, ya sea decimal o científica
%c entrada de una variable tipo char
%s entrada de una cadena
%o entrada de un número octal
%x entrada de un número hexadecimal

3.4.4.1. Flujo de Entrada

La función scanf() utiliza los especificadores de formato para interpretar cómo se deberá utilizar los caracteres obtenidos.

Los especificadores de formatos se denominan caracteres de conversión, por que convierten en datos los caracteres recibidos en el flujo de entrada, basándose en los tipos especificados.

Scanf( ) de manera automática ignora todos los caracteres de espaciamiento (espacios), tabuladores o comandos de nueva línea.

```
main( )
{
        int contador;
        puts("Por favor ingrese un número:");
        scanf("%d", &contador);
        printf("El número es %d", contador);
}
```

Y despliega la siguiente información en la pantalla

Por favor ingrese un número:
El número es X

Se puede pulsar la barra espaciadora tantas veces como se desee antes de teclear el número. C ignorará los espacios y buscará el primer carácter no blanco.

Sin embargo la asignación se suspende con el primer carácter no coincidente o no numérico. Por lo tanto, si se ingresó 123abc, se asignará el valor 123 a la variable y las letras abc se ignorarán.

La asignación también se suspenderá con el primer carácter espacio – si se ingresa 12 3, se asignará el valor 12 a la variable y el número 3 se ignorará.

Cuando se ingresa una cadena scanf() inicia la asignación de valor con el primer carácter diferente a un espacio. Se suspenderá la asignación de valor con el siguiente carácter espacio. Ejemplo:

```
main()
{
        char nombre[25];
        puts ("Por favor proporcione su nombre: ");
        scanf ("%s", nombre);
        puts  (nombre);
}
```

Observe que no se utilizó el *ampersand* asociado al nombre de la variable, por ser tipo cadena, además con cualquier espacio en blanco una vez iniciado el ingreso se suspende la asignación, es decir si ingresara *Ulises Olivares*, solo se asignará el nombre de Ulises y se ignorará el apellido *Olivares.* Debido a la manera en que opera scanf(), si desea ingresar una cadena con espacios, se deberá utilizar la función gets().

3.5. OPERADORES

La fase del procesamiento de un programa es convertir los datos que reingresan en información a la que se le da salida. La diferencia entre datos e información es sutil, pero importante. Los datos son materia prima, caracteres y números que son valiosos pero en una forma que no permite utilizarlos como un producto terminado. La información es el producto terminado; su obtención es el motivo por el que se escriben programas.

3.5.1. ORDEN DE PRECEDENCIA

Cuando la computadora encuentra una expresión, no realiza el cálculo de izquierda a derecha de manera automática. Primero analiza la línea y ejecuta las operaciones matemáticas en base al orden de precedencia de los operadores. Orden de precedencia significa que algunas operaciones se realizan antes que otras, aunque aparezcan más adelante en la ecuación.

La computadora asignará ubicaciones de memoria de manera temporal para almacenar los resultados, después regresará al principio de la ecuación y ejecutará las adiciones y restas en el orden en que las encuentre.

Considere esta sencilla ecuación:

a = 1 + 4 / 2 * 5 + 3

Si se efectuar el cálculo simplemente de izquierda a derecha, como cuando se utiliza una calculadora el resultado es de 15.5. Sin embargo la computadora realiza la operación respetando el orden de precedencia como lo indica la siguiente figura.

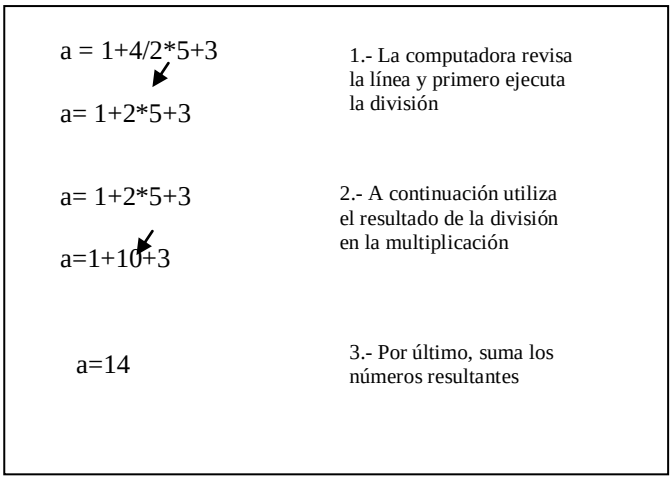


FIGURA 2.11. EJEMPLO DEL ORDEN DE PRECEDENCIA.

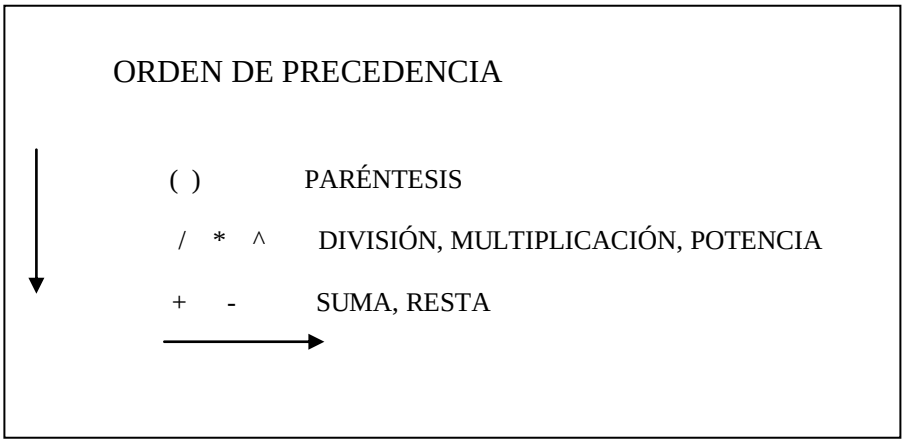


FIGURA 2.12. ORDEN DE PRECEDENCIA.

Como se vio en el ejemplo anterior, una variable aparecerá siempre del lado izquierdo de un signo de igualdad, y el operador aritmético en el lado derecho, ejemplos:

impto_venta= importe*tasa_impto;

precio=costo+ flete+ seguro;

premedio=(cali_1+cali_2+cali_3+cali_4)/4;

3.5.2. OPERADORES Y TIPO DATOS.

Normalmente se utiliza el mismo tipo de dato (int o float) en ambos lados del signo igual en una operación aritmética. Por ejemplo, si se están sumando dos valores float se asignará el resultado una variable float.

Sin embargo si se realiza una operación donde se suman dos variables tipo float (por ejemplo costo y fletes) y el resultado se asigna una variable tipo int. Debido a que el resultado esta declarado como un int, el resultado será un int es decir un entero, como se indica en el siguiente ejemplo.

```
main()
{
        int total;
        float costo, fletes;
        costo=56.09;
        fletes=4.98;
        total= costos + fletes;
        printf("El total de su fatura es $%d",total);
}
```

Gracias al especificador de formato %d desplegara en pantalla el siguiente resultado:

El total de su factura es $61

Ignorando la parte decimal es decir 61.07 ya que el total no fue declarado como float (decimal).

3.6. CONTADORES

Un contador es una variable que tiene un valor inicial que se ve incrementado por uno cada vez que algo sucede. El algoritmo para un contador es:

variable = variable+1

Un maestro de matemáticas sin duda objetaría esta fórmula, reclamando el hecho de que no se puede tener la misma variable en ambos lados de una ecuación. Sin embargo, esto es perfectamente legal en el ámbito de la programación de computadoras.

La computadora evalúa (calcula) primera la expresión del lado derecho de la igualdad y después asigna el resultado a la variable especificada en el lado izquierdo. En ningún momento la variable tiene más de un valor. Hay que considerar un contador de la siguiente manera:

El nuevo valor de la variable es igual al valor anterior más 1.

Por supuesto, se puede incrementar por cualquier número y empezar con un valor inicial cualquiera. Si se asigna la variable contador un valor inicial de 1, la instrucción:

contador = contador + 2

Lo incrementará dándole los valores de los números nones – 1,3,5,7, etc. Se puede contar de cinco en cinco utilizando contador = contador + 5; o con incrementos de diez con:

contador = contador + 10; o con cualquier otro número.

La cuenta inversa (decremento) sólo requiere cambiar el algoritmo a:

variable = variable – 1

cada vez que se realice la operación, el valor de la variable se decrementará en 1.

3.6.1. Operadores de Incremento

Los contadores se utilizan con tanta frecuencia que C proporciona operadores especiales para incrementar y decrementar una variable. El operador ++variable suma 1 al valor que tenga la variable antes de ejecutar la instrucción. La operación ejecuta la misma operación que

variable = variable + 1;

Como ejemplo véase el siguiente programa:

```
/*conteo.c*/
main( )
{
        int contador = 0;
        printf("El primer valor del contador es %d\n, contador);
        printf("El segundo valor  es %d\n, ++ contador);
        printf("El último valor fue %d\n", contador);
}
```

La salida del programa es:

El primer valor del contador es 0
El segundo valor es 1

El último valor fue 1

Es importante que se entienda la diferencia entre el operador de incremento ++ contador y la expresión contador + 1 en la siguiente línea:

printf("El segundo valor es % d\n, contador + 1);

La expresión contador +1 no modifica el valor de la variable contador. Sólo da lugar a que se despliegue un valor incrementado. Este ejemplo muestra un programa que utiliza esta expresión.

3.6.1.1. Empleo de una expresión en lugar del operador de incremento

```
main( )
{
        int contador = 0;
        printf("El primer valor del contador es %d\n, contador);
        printf("El segundo valor  es %d\n,  contador +1);
        printf("El último valor fue %d\n", contador);
}
```

El programa dará lugar a la siguiente salida:

El primer valor del contador es 0
El segundo valor es 1
El último valor fue 0

La expresión contador +1 se evalúa como 1, pero no modifica el valor de la variable contador. Cuando la variable se despliega por la tercera instrucción printf( ) todavía tiene su valor original.

Si se colocan los signos ++ en el otro lado del nombre de la variable se incrementará el valor de ésta después de haber sido complementada la instrucción. La sintaxis es:

variable + +

Los operadores de decremento funcionan de la misma manera, pero reducen el valor de la variable en uno. Su sintaxis es

--variable Decrementa en uno el valor antes de que se ejecute la instrucción.
variable-- Decrementa en uno el valor después de que se ejecuta la instrucción.

3.7. ACUMULADORES

Un acumulador también incrementa el valor de una variable. Pero en lugar de incrementa el valor siempre por la misma cantidad el incremento es variable – puede cambiar con cada operación. La sintaxis general es:

variable = variable + otra _ variable

Se le llama acumulador por que el valor se va acumulando.

La siguiente lista muestra un programa que solicita tres números y calcula su promedio. Se pudo haber utilizado una expresión matemática simple para calcular el promedio:

Promedio = (A+B+C)/3

Se ha usado un acumulador para totalizar los números y un contador para indicar las veces que se repitió el evento.

Ejemplo: Programa que emplea un contador y un acumulador para calcular el promedio de tres números

```
/*promedio.c*/
main( )
}
{
float numero, total, contador, promedio;
total = 0.0;
contador = 0.0;
printf(Proporcione el primer número:”);
scanf(%f”, &numero);
total + = numero;          /*indica la siguiente instrucción total=total+número*/
++contador;                /*indica la siguiente instrucción contador=contador + 1*/
printf(“Ahora, el segundo número:”)
scanf(“%f”, &numero);
total += numero;
++contador
printf(“¿Valor del tercer número?:”);
scanf(“%f”, &numero);
total += numero;
++contador
promedio =total/contador;
printf(“El promedio es %f”,promedio);
}
```

3.7.1. Operadores de Asignación

Los operadores de asignación son abreviaturas de acumuladores:

Operador	Ejemplo	Equivalente
+=	total += importe	total = total + importe
-=	total -= descuento	total = total – descuento
*=	total*= tasa_imp	total=total* tasa_imp
/=	total/= contador	total=total/ contador
%=	total%= contador	total=total% contador

3.8. FUNCIONES

Una función es un bloque de instrucciones que realizan una tarea específica. Se escriben funciones propias y se utilizan de la misma manera que las funciones en la biblioteca C o C++ – se llama a la función para realizar el trabajo, incluso pasando y recibiendo de ella argumentos.

3.8.1. Uso de funciones

Las funciones se colocan después del corchete de cierre correspondiente a main( ). Cada función tiene la misma estructura que main( ), tal como se muestra en la figura siguiente. Tal y como main( ), el nombre de la función deberá ir seguido por paréntesis, sin punto y coma final.

El comando return(0) envía el control de regreso a la instrucción inmediata al llamado a la función.

Una función puede ser llamada desde cualquier lugar del programa, aun desde dentro de otra función.

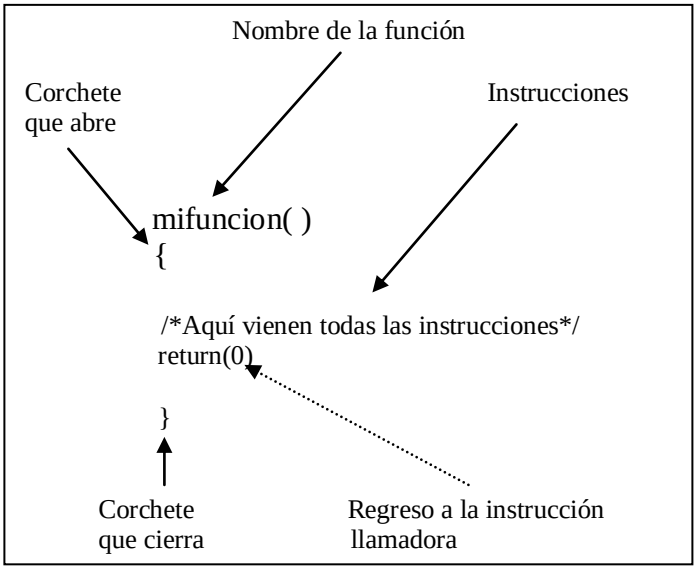


FIGURA 2.13. USO DE FUNCION.

Ejemplo: Programa con main( ) que ejecuta los llamados a todas las funciones.

```
/*examen3.c*
main( )
{
    saludo( );
    pregunta( );
    respuesta( );
    el_final( );
}
saludo( )
{
    puts("Bienvenidos a la competencia Sybex.\n");
}
pregunta( )
{
    int sigue
    puts("Dé el nombre de una interfaz gráfica de Microsoft.\n");
    puts("Presione Enter para conocer la respuesta.\n");
    sigue = getchar( );
}
respuesta( )
    puts("La respuesta es Windows.\n);
{
```

```
el_final( )
    puts(“Gracias por su patrocinio.\n);
}
```

Las funciones se realizan en orden en que son llamadas, no como aparecen en el programa.

3.8.2. Variables Externas (Globales)

Una variable externa es aquella que puede ser empleada por cualquier función en el programa. Algunos lenguajes llaman globales a este tipo de variables. Para que sea externa, la variable deberá ser declarada antes de la función main( ), como en:

```
int temp;
main( )
```

Aquí la variable temp es externa y, por lo tanto, podrá ser usada en cualquier función dentro del programa.

Si existen una variable externa y una local con el mismo nombre, en la función se usará la variable local.

3.8.3. Paso de Parámetros

Cualquier dato que se quiera enviar a una función deberá estar entre los paréntesis. La lista de variables pasadas a una función se conoce como lista de argumentos.

Ejemplo: Paso de parámetros

```
voit area(float,float,int)
main( )
{
     float largo, ancho;
     int piso;
     printf(Proporcione el número del piso:”);
     scanf(“%d”,&piso);
     printf(“Ingrese el largo del piso:”);
     scanf(“%f”,&largo);
     printf(“y su ancho:”);
     scanf(“%f”,&ancho);
     area(largo, ancho, piso);

area(valargo, vaaancho, nivel)
float valargo, vaancho;          voit area (float valargo,
                                 float banco, int nivel)
```

```
int nivel;
      float area;
      area=valargo*vaancho;
      printf("El área del piso %d es de %.2f", nivel, area);
      return(0);
}
```

Los tres argumentos se reciben en el mismo orden en que se pasan o procesan.

Los compiladores C++ y algunos C recomiendan que se inicie el programa con los prototipos de funciones. Un prototipo en sólo la línea de declaración de la función repetida al inicio del programa, antes de main( ), como en:

```
main( )
```

El prototipo notifica al compilador respecto al tipo de funciones y argumentos que se emplearán en el programa.

3.9. CONDICION

3.9.1. if

Se puede emplear el comando if para decir sin una instrucción debe de ejecutarse. La estructura del comando es:

```
if(condición)
   instrucción;
```

El comando quiere decir "si esta condición es verdadera, entonces ejecuta la siguiente instrucción". La computadora ejecuta la instrucción y después continua de manera normal, ejecutando las instrucciones que siguen al comando if.

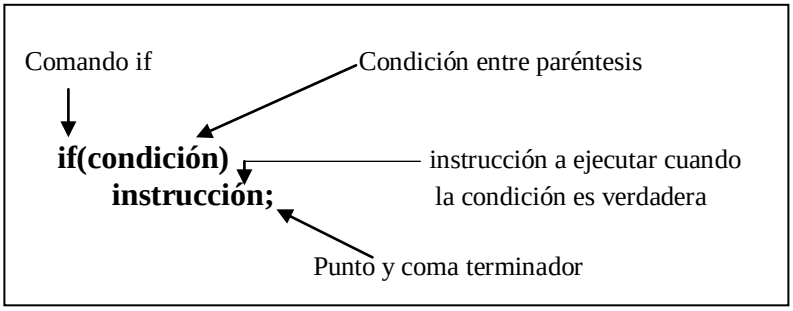


FIGURA 2.14. USO DEL LA CONDICIÓN if.

Escribir un if es sencillo siempre y cuando se recuerden los siguientes puntos básicos:

1-. Se debe colocar la condición entre paréntesis.
2-. No debe incluirse el punto y coma después de la condición
3-. Deberá aparecer después de la condición.

C es un leguaje de formato libre, por lo tanto, la condición y la instrucción puede estar en la misma línea. No obstante, separar e identificar la condición hace mas fácil la lectura del programa.

3.9.2. Condiciones

Las condiciones en los comandos if comparan valores – una variable – una variable o constante.

El signo igual sencillo = se reserva para asignar valores a las variables.

En la siguiente figura indica el símbolo, así como su significado y además un ejemplo de cómo utilizar con cada comparador

Operador	**Significado**	**Ejemplo**
= =	igual a	if (impuesto = = 0.06
>	mayor que	if (horas>40)
<	menor que	if (horas<40)
>=	mayor que o igual a	if (sueldo>=1000)
< =	menor que o igual a	if (costo< =limite)
!=	diferente	if (contador !=1)

FIGURA 2.15. COMPARADORES PARA LA CONDICION if.

3.9.3. Enunciados Múltiples

El enunciado if básico ejecuta una instrucción. Si se quiere ejecutar más de una instrucción cuando la condición sea verdadera, se deberán emplear un segundo nivel de corchetes. El corchete después de la condición marca el principio de un bloque, y el corchete después de la última instrucción condicionada marca el final del bloque.

Cualquier instrucción dentro del bloque se ejecutará si la condición es verdadera.

Ejemplo: Enunciados múltiples

```
/*lujo2.c*/
main( )
{
      float costo, impuesto, lujo, total
      lujo=0.0;
      prinf("Proporcione el costo del artículo:");
      scanf("%f", &costo);
      impuesto = costo * 0.06;
      if(costo>40000.00)
     {
        total=costo+impuesto+lujo;
        printf("El costo total es %.2f",total);
}
```

3.9.4. El Comando if…else

Cuando se utiliza el comando if solo se está interesado en ejecutar instrucciones cuando la condición es verdadera. También puede ejecutarse un conjunto de instrucciones cuando la condición es verdadera y otro conjunto cuando la condición es falsa.

Las instrucciones se combinan usando el comando else.

```
if(condición)
      instrucción;
else
      instrucción;
```

la estructura dice "si la condición es verdadera, ejecuta la siguiente instrucción; en caso contrario ejecuta la instrucción que sigue al comando else". Las instrucciones que siguen a else se ejecutan solo si la condición es falsa. Si en cualquiera de los casos se requiere ejecutar mas de una instrucción, estás deberán ir entre corchetes. Se requiere un punto y coma después de cada instrucción, pero no después de la palabra clave else.

3.9.5. Operadores Lógicos

Las condiciones if estudiadas hasta este punto prueban una sola variable contra un valor. Esto es, sólo se deberá satisfacer una condición para que el enunciado sea evaluado como verdadero. En el mundo real, a menudo los programas necesitan hacer la prueba por más de un valor.

Existen tres operadores lógicos: ¦ ¦ para "O"(OR),&&para "Y"(AND), y ! para "NO"(NOT). Cuando se utiliza el operador OR, cuando menos una de las condiciones deberá ser

verdadera para que el enunciado if se considere verdadero. Con el condicionante AND ambos deberán ser verdaderos. Cuando se usa el operador NOT la condición deberá ser falsa.

Por ejemplo, supóngase que se requiere un programa que solicita por el telado el ingreso mensual del usuario y el número de sus dependientes, como en el ejemplo siguiente.

Ejemplo: Prueba de dos variables

```
main( )
{
        int dependie;
        float ingreso;
        puts("Por favor, proporcione su ingreso mensual");
        scanf("%f", &ingreso);
        puts("y el número de sus dependientes");
        scanf("%d", &dependie);
        if(ingreso < 20000.00 &&dependie > 2)
        puts("No adeuda impuestos por ingresos");
}
```

La condición if prueba dos variables, ingreso y depende. El impuesto deberá ser menor que $20,000 Y el número de dependientes deberá ser mayor que dos para que se despliegue el mensaje. Si cualquiera de las condiciones es falsa, tal como un solo dependiente o ingreso por $20,001, la instrucción puts( ) no se ejecutará.

3.9.6. Enunciados if Anidados

Una condición if o un comando else puede ser seguido por cualquier tipo de instrucción. Dicha instrucción puede ingresar o dar salida a un valor, realizar una operación matemática o llamar a una función específica. La instrucción también puede ser otro comando if. Cuando un comando if está contenido dentro de otro, se dice que esta anidado. Por ejemplo, en está instrucción el segundo comando if está anidado dentro de la primera:

```
if(ingreso > 100000.00)
   if (estado = = 'S')
     tasaimp = 0.35;
```

La segunda condición if será probada sólo cuando la primera condición sea verdadera, por lo tanto, se asignará el valor de la variable tasaimp sólo cuando ambas condiciones sean verdaderas. La misma lógica se puede escribir como:

```
if(ingreso > 100000.00 && estado = = 'S')
   tasaimp = 0.35;
```

3.10. REPETICIÒN.

C y C++ tienen tres diferentes estructuras, conocidas como repeticiones o ciclos para controlar la repetición:

- el ciclo for
- el ciclo do…while
- el ciclo while

Cualquiera de ellos puede repetir una instrucción, un grupo de instrucciones o un programa completo.

3.10.1. Uso del Ciclo for

Se emplea el ciclo for cuando se conoce el número exacto de repeticiones que se desea ejecutar. La estructura del ciclo se ilustra en la siguiente figura.

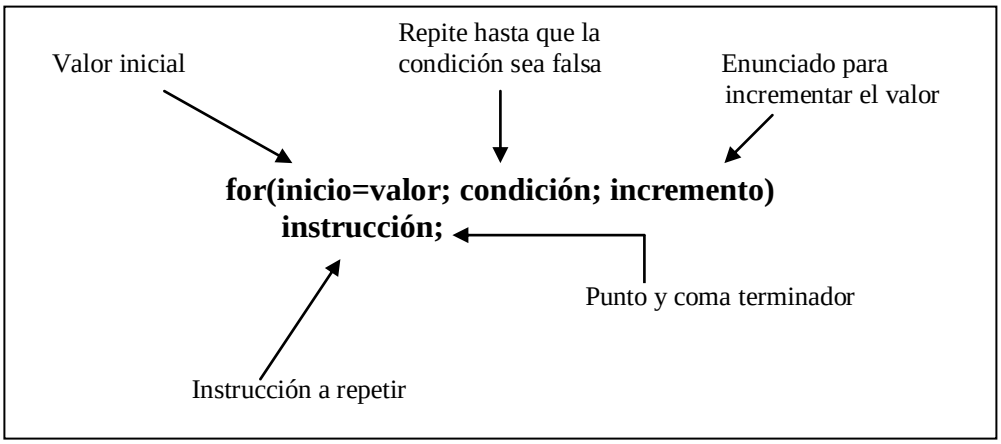


FIGURA 2.16. USO DEL CICLO for.

3.10.1.1. Instrucciones Múltiples

Para ejecutar más de una instrucción en un ciclo, se coloca todo el conjunto de ellas entre los corchetes. Por ejemplo, el siguiente es el programa que convierte 101 temperaturas consecutivas (de 32 a 132) a grados Celsius:

```
main( )
{
        int temper;
        float Celsius;
          puts(“Fahrenheit\Celsius\n”);
          for(temper=32; temper<=132; temper++)
          {
                celsius = (5.0/9.0)*(temper-32);
                printf(“%d\t\t%6.2f\n”, temper, celsius);
           }
}
```

3.10.1.2. Uso de Variables

Si al momento de escribir el programa no se conoce el número de repeticiones requeridas, aún es posible utilizar el ciclo for si dicho número se puede conocer al momento de la ejecución del programa. Puede ingresarse un valor en la variable y emplearlo en una condición. Por ejemplo, este programa solicita al usuario le proporcione un rango de temperaturas a convertir – de hecho, el número de veces que se desea repetir una parte del programa:

```
main( )
{
      int temper, inicio, final;
      float celsius;
          printf(“Proporcione la temperatura inicial:”)
          scanf(“%d”, &inicio);
          printf(“Proporcione la temperatura final:”)
          scanf(“%d”, &final);
          puts(“Fahrenheit\Celsius\n”);
          for(temper=inicio; temper<=final; temper++)
           {
                celsius = (5.0/9.0)*(temper-32);
                printf(“%d\t\t%6.2f\n”, temper, celsius);
            }
}
```

3.10.1.3. Ciclos anidados

Cuando se ejecuta un comando for dentro de otro, se dice que están anidados. El ciclo interno se repite completamente por cada repetición del ciclo externo. Se puede considerar a los ciclos for anidados como bidimensionales y los ciclos for sencillos como uni-dimensionales.

Ejemplo:

```
main( )

{
int fila, columna;
for(fila=1; fila<=10; columna++);
        {
        printf(“*”);for(columna=1; columna<=10; columna++);
        putchar(‘\n’); /*fuera del segundo ciclo, dentro del primero*/
        }
}
```

Por ultimo considere el siguiente programa:

```
main( )

{
int fila, columna;
++++++++++++
for(fila=1; fila=10; fila=++)
    {
    for(columna=1; columna<=fila;columna++)
     printf(“*”);
    putchar(‘\n’);
    }
}
```

Este programa desplegará la serie de asteriscos mostrados en la siguiente figura:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

FIGURA 2.17. ASTERISCOS UTILIZANDO EL for.

3.10.2. Uso del ciclo do…while

El ciclo do…while se emplea cuando no se conoce el número exacto de repeticiones a ejecutar, pero cuando menos se desea ejecutar el ciclo una vez. La estructura del comando se ilustra en la siguiente figura.

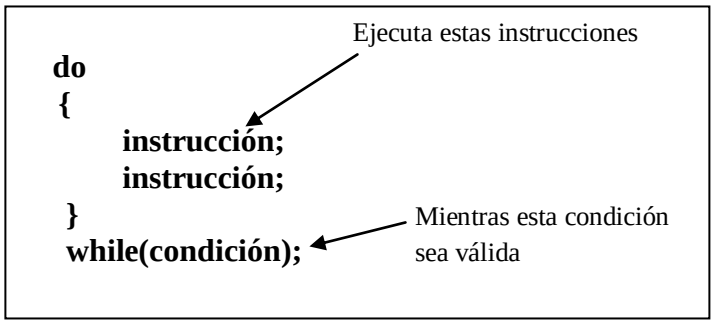


FIGURA 2.18. USO DE CICLO do…while.

Ejemplo:

```
main( )
{
    int contador;
    flota descto;
    contador=0;
do
     {
      printf("Por favor proporcione el descuento:");
      scanf ("%f", &descto);
      contador + +
      }
         white(descto<=0 || descto>=1)&&contador<20);
         if(contador = = 20) puts("Ignorante");
}
```

3.10.3. Uso del Ciclo while

Este ciclo se emplea, cuando no se conoce el número de repeticiones e incluso, puede ser que no se requiera ejecutar el ciclo.

A diferencia de la estructura do, la condición será probada incluso la primera vez, antes de que se ejecute el ciclo. Si la condición es falsa el ciclo no se ejecutará, ni siquiera una vez.

Una ilustración de la sintaxis aparece en la figura 2.19. Para instrucciones múltiples se usa este formato:

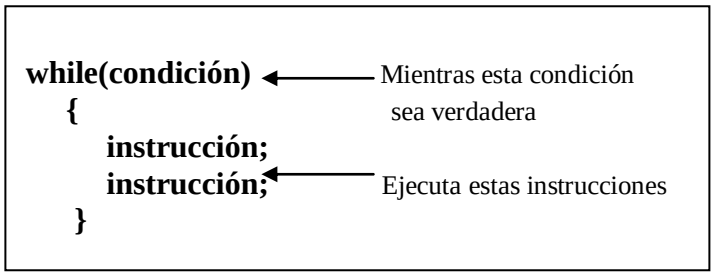


FIGURA 2.19. USO DE LA FUNCIÓN while.

3.11. ARREGLOS

Un arreglo es una colección de valores que pueden operarse como un grupo.

3.11.1. Declaración de un Arreglo

Se declara un arreglo al asignar el tipo de valores que almacenará y el número máximo de elementos, empleando la sintaxis mostrada en la figura. Por ejemplo, para crear un arreglo de treinta y uno números enteros, la cual se presenta por medio de la siguiente sintaxis.

int temps[31];

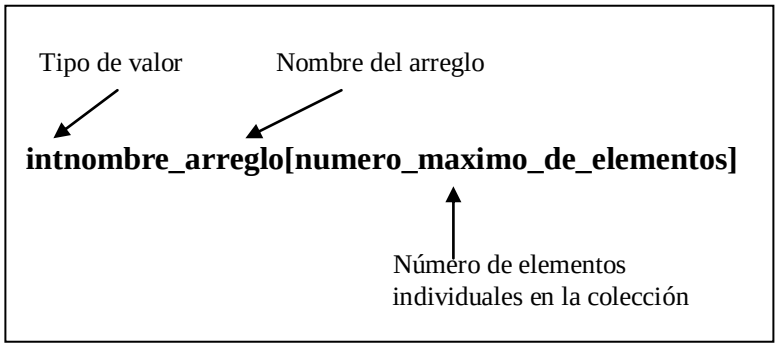


FIGURA 2.20. SINTAXIS DE UN ARREGLO.

3.11.2. Manejo de Arreglos

Los valores almacenados en un arreglo podrán ser accesados siempre que se requieran en el programa. En la siguiente lista, por ejemplo: utilizar un arreglo para dos diferentes tareas.

Ejemplo: Utilización de un arreglo para dos tareas

```
main( )
{
        int temps[31];
        int indice, total;
        float promedio, celsius;
        total=0.0;
        /*carga de arreglo con valores*/
        for(indice=0; indice<31; indice++)
        {
            printf("Proporcione el valor de la temperatura #%d:", indice);
            scanf("%d", &temps[indice];
        }
        /*lectura de los valores para calcular el promedio*/
        for(indice=0; indice<31; indice++)
            total+=temps[indice];
        promedio=total/31.0;
        printf("El promedio es: %f\n\n", promedio);
        puts("Fahrenheit\Celsius\n");
        /*lectura de los valores para conversión*/
        for(indice=0; indice<31; indice++)
        {
          celsius=(5.0/9.0)*(temps[indice]-32);
          printf("%d\t\t%6.2f\n", temps[indice]-celcius);
        }
}
```

Ejercicio Sobre Arreglos

El ejemplo siguiente muestra la forma en como se utilizan los arreglos bidimensionales.

Existen diez salones para juntas, cada uno con una capacidad legal específica. Se desea tener un programa que ejecute las siguientes tres funciones:

- Despliegue una tabla mostrando el número de cada salón y su máxima ocupación autorizada.
- Despliegue la ocupación máxima autorizada de un salón en especial.
- Despliegue una lista de los salones que tienen una ocupación máxima autorizada específica.

Existen varias maneras de construir un programa para este caso.

Una alternativa es emplear dos arreglos paralelos. Los arreglos paralelos independientes que se pueden relacionar. El programa en la lista muestra una solución para el problema de administración de salones utilizando arreglos paralelos. El arreglo salones contiene una lista de los diez números de los salones que están disponibles. El arreglo máximo contiene los límites de ocupación de los diez

salones. Si se localiza el número del salón en el elemento 5 del arreglo salones, se sabe que su límite de ocupación está en el elemento 5 del arreglo máximo.

Ejemplo: Empleo de los dos arreglos paralelos

```
int salones[10]={102, 107, 109, 112, 115, 116, 123, 125, 127,130};
int maximos[10]={12, 43, 23, 12, 20, 15, 16, 23, 12, 15};
main( )
{
int indice, selecc, numero, limite, bandera, hubo;
limites=10;
puts("1. Capacidad por un número de salón\n");
puts("2. Capacidad de un salón específico\n");
puts("3. Localizar salones de cierta capacidad\n");
printf("Haga su elección (1 a 3):");
scanf("%d", &selecc);
if(selecc= =1)
{
    for(indice=0;indice<limite;indice++)
    printf("Salón %d %d máximo\n", salones[indice], maximos[indice];
}
if(selecc= =2)
{
    printf("Proporcione el número del salón:");
    scanf("%d", &numero);
    indice=1;
    hubo=0;
    while(!hubo && indice<limite)
        if(salones[indice]= = numero)
            hubo=1;
        else
            indice=++;
    if(hubo)
        puts(*Ese número de habitación no es para este tipo de uso\n");
        else
printf("El salón %d tiene capacidad para %d personas\n",
salones[indice], maximos[indice]);
}
if(selecc= =3)
{
        banderao=0;
           printf("Proporcione la capacidad mínima requerida:");
           scanf("%d", &numero);
           for(indice=0; indice<limite; indice++)
             if(maximos[indice]>=numero)
                 {
                  Bandera=1;
printf("El salón %d tiene capacidad para %dpersonas\n",salones[indice],
```

```
maximos[indice];
}
        if (bandera= =0)
                puts("No hay salones con tal capacidad\n);
}
}
```

El programa asigna el valor máximo del índice para el arreglo a la variable límite, y después despliega un menú de opciones. La selección determinada que función se debe de realizar por medio de un grupo de tres enunciados if. El programa también se pudo haber escrito empleando en su lugar un enunciado switch o instrucciones if…else anidadas.

3.12. CADENAS

Una cadena es un arreglo de caracteres. Cuando se declara una cadena se le da un nombre y se específica el número máximo de caracteres que pueden contener. Sin embargo, el último elemento del arreglo se reserva para el terminador nulo (\0), por lo tanto, se requiere declarar el arreglo con un número de elementos igual al número máximo de caracteres más uno. Por ejemplo, se puede ingresar una cadena y después desplegar los caracteres individuales que la forman, de la siguiente manera:

```
main( )

{
char nombre[20];
int indice;
```

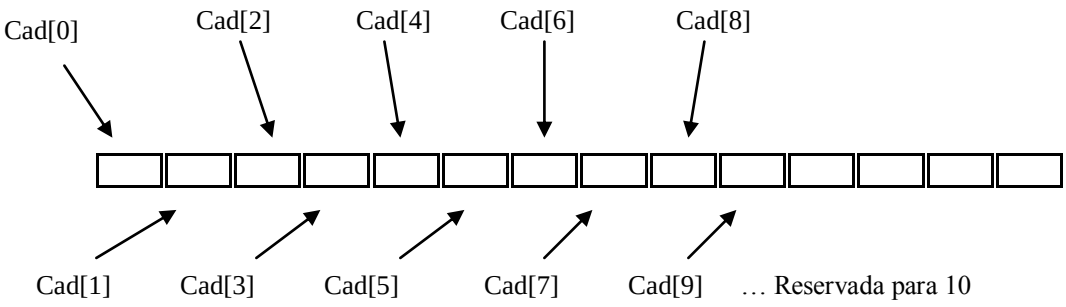


FIGURA 2.21. ESTRUCTURA DE UNA CADENA.

```
printf("Proporcione su nombre;");
scanf("%s", nombre);
for (indice=0; indice<20, indice++)
```

```
 printf("%c\n", nombre[indice]);
}
```

Si no se ingresan los 20 caracteres en su totalidad, aquellos después del terminador nulo tendrán valores desconocidos.

3.12.1. Arreglos de Cadenas

Se puede tener un arreglo de cadenas, tal y como se pueden tener arreglos de cualquier otro tipo de datos. Un arreglo de cadenas es, en realidad, un arreglo de arreglos de caracteres. A un arreglo cuyos elementos son arreglos se le llama arreglo de dos dimensiones o bidimensional.

Se puede imaginar un arreglo bidimensional como una hoja tabular con filas y columnas. Se debe definir un arreglo así mediante dos índices:

Uno especifica el número de filas o renglones en la hoja tabular, y el otro representa el número de columnas.

La siguiente instrucción declara un arreglo de 10 filas y 20 columnas, o sea 200 variables enteras en total:

```
int tabla[10] [20]
```

Cuando se declara un arreglo de cadenas también se debe utilizar dos índices. El primero representa el número máximo de cadenas en el arreglo; el segundo representa la longitud máxima de cada cadena. Por lo tanto, si se declara:

```
char nombres[10] [20];
```

Se podrá tener hasta diez nombres de hasta 19 caracteres cada uno como lo muestra la figura 2.22.

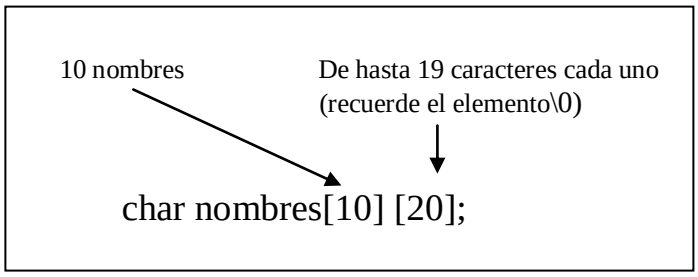


FIGURA 2.22. ARREGLO PARALELO.

3.13. APUNTADORES

Si se asigna a una variable un valor, tal como:

impuesto=35;

Se estará introduciendo un valor en una posición de memoria. La dirección de memoria asignada a la variable impuesto contendrá el valor 35. También se puede desplegar la dirección en memoria de una variable utilizando el operador &. La instrucción

printf("%d," &impuesto);

desplegara la dirección donde esta almacenada la variable impuesto, y no al valor almacenado en dicha dirección. Sin embargo, el operador de dirección, como & impuesto, sólo se puede utilizar en expresiones. No es una variable, ya que sólo representa a un número entero; una instrucción como

&impuesto=25;

es válida.

Un apuntador es una variable que contiene la dirección en memoria de otra variable. Si la variable impuesto esta almacenado en la dirección 21260, entonces un apuntador a la variable impuesto contendrá 21260.

Para crear una variable apuntador se emplea una sintaxis mostrada en la figura 2.23. Se inicia especificando el tipo del dato almacenado en la dirección identificada por el apuntador. El asterisco le indica al compilador, que se está creando una variable apuntador. Por último se da el nombre de la variable. Por ejemplo,

int *impuptr

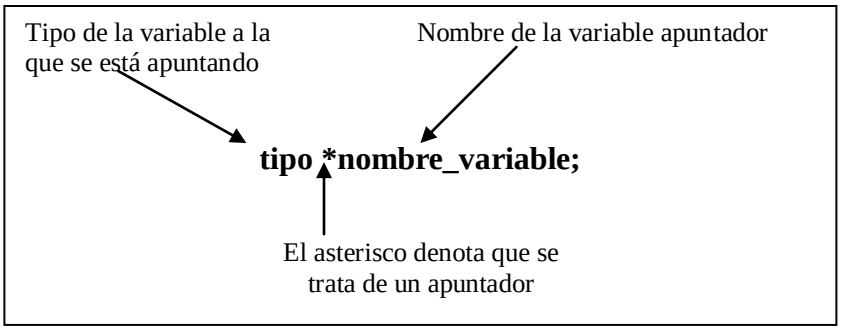


FIGURA 2.23. SINTAXIS DEL APUNTADOR.

El beneficio es que se puede hacer referencia a la variable apuntador como *impuptr. El asterisco le indica al compilador que se esta interesando en el contenido de la dirección de memoria almacenada el apuntador.

No obstante, sí se puede asignar un valor al apuntador *impuptr, como:

*impuptr=35

Esto significa "coloca el valor 35 en la dirección de memoria a la qua apunta a la variable impuptr". Debido a que el apuntador contiene la dirección 21260 el valor 35 se almacenará en dicha posición de memoria. Y como esta es la dirección del la variable impuesto, el valor de impuesto también será 35 observe la figura 2.24.

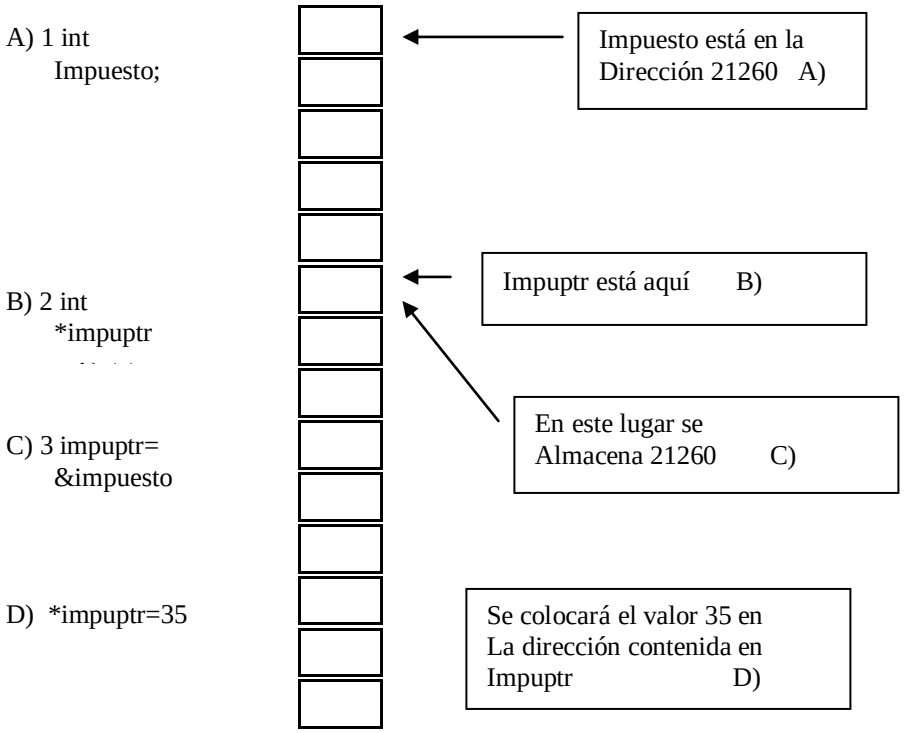


FIGURA 2.24. FUNCIÓN DE UN APUNTADOR.

La potencialidad de los apuntadores radica en su uso como argumentos en funciones.

CAPÍTULO 4

EXPERIMENTOS ELÉCTRICOS

4.1. INTRODUCCIÓN

La electricidad es la más flexible y versátil de todas las formas de energía existentes. ¿Quién no conoce sus numerosas aplicaciones en dispositivos electromecánicos tanto caseros, como industriales, ya sea para alumbrado, calefacción, propulsión de motores entre otros.

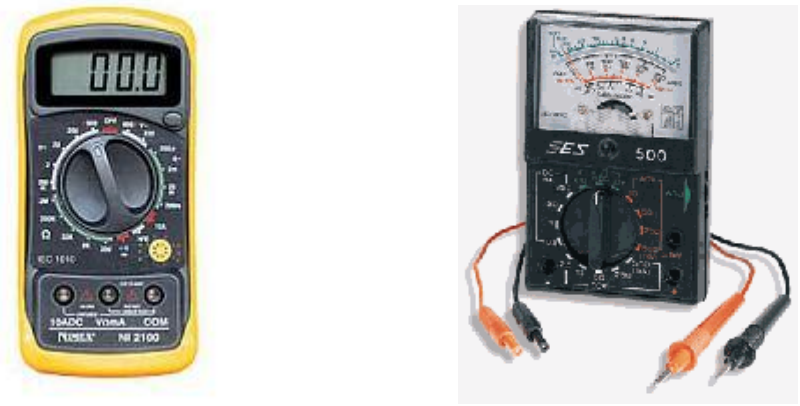
Desde los tiempos de Volta, inventor de la primera pila eléctrica, y Tesla, quién ideo el primer motor de inducción, se han logrado grandes progresos en el aprovechamiento de la electricidad para suplir las energías y fuerza del hombre. Los generadores que convierten la energía mecánica en eléctrica y que se construyen actualmente, pueden producir más de 500,000 Kilowatts. En cuanto a la conversión de la energía eléctrica en trabajo mecánico, hay una sombrosa variedad de motores que van desde el diminuto propulsor de una rasuradora eléctrica hasta el de un gigantesco ventilador de 20,000 caballos de potencia que impulsa aire a través de un túnel aerodinámico (túnel de viento).

Por lo tanto no es sorprendente que la energía eléctrica ocupe un lugar preponderante en todos los niveles de estudio en la tecnología industrial.

4.2. RESISTENCIAS EQUIVALENTES EN SERIE Y EN PARALELO

Todos los materiales poseen resistencia eléctrica (oposición al flujo de la corriente eléctrica) en mayor o menor grado. Los materiales tales como la plata, el cobre y el aluminio, que contiene una resistencia relativamente baja, se conocen con el nombre de **conductores**, tanto que los materiales tales como los plásticos, el vidrio, el aire y el caucho que tienen una resistencia muy alta se denominan **aisladores**. Entre estas dos categorías principales existe una gran variedad de materiales y aleaciones cuya resistencia no es muy alta ni muy baja. No se puede trazar una línea divisoria claramente definida entre los conductores y los aisladores. Los conductores gradualmente se convierten en resistencias y éstas en aisladores. Todos los materiales, incluyendo los conductores, tienen resistencia eléctrica. Se dice que un material tiene poca resistencia eléctrica cuando ofrece una oposición débil al paso de la corriente eléctrica. La unidad de la resistencia es el **ohm**.

Uno de los instrumentos para medir la resistencia es el **óhmetro** u **ohmímetro**. Por lo general los ohmímetros se componen de una fuente de voltaje de c-d (usualmente, una batería), un medidor de corriente y un conmutador de rangos o de gamas para seleccionar las resistencias de calibración interna. La escala del medidor se calibra en función del valor de la resistencia que produce una corriente dada. La resistencia desconocida se conecta entre las terminales (cables) del ohmímetro y la señala en la escala el valor de la resistencia.



b) Multimetro Digital. a) Multimetro Analógico.

FIGURA 4.1. MULTIMETROS.

Los circuitos en serie y los de paralelo se calculan con facilidad, cuando menos en lo que respecta a la resistencia equivalente. Los circuitos conectados en serie-paralelo no son necesariamente más difíciles y el secreto esta en reducir los diversos elementos de circuito a valores en serie y en paralelo hasta que todo el circuito se haya cambiado a una sola resistencia equivalente.

4.3. RESISTENCIAS EN SERIE

Cuando un grupo de resistencias se conectan en serie, la resistencia total es igual a la suma de los valores de cada una de las resistencias. Por lo tanto, si una resistencia que tiene un valor de 5 ohms se conecta en serie con otra que tiene un valor de 20 ohms la resistencia total entre las terminales A y B es de 25 ohms, como se muestra en la siguiente figura.

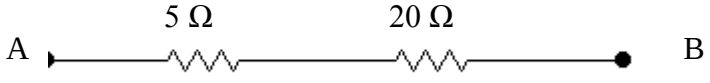


FIGURA 4.2. RESISTENCIAS EN SERIE.

Las dos resistencias (R_1 y R_2) entre las terminales A y B se pueden sustituir por una sola (R_3) que tenga un valor de 25 ohms. Esta sola resistencia (R_3), que puede sustituir a las dos originales se denomina **resistencia equivalente** (figura 4.3).

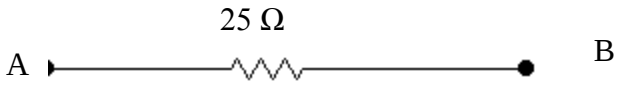


FIGURA 4.3. RESISTENCIA EQUIVALENTE.

La resistencia equivalente a varias resistencias conectadas en serie se encuentra mediante la ecuación:

$$R_{equivalente} = R_1 + R_2 + R_3 + \ldots\ldots\ldots \quad (1)$$

4.4. RESISTENCIAS EN PARALELO

Cuando dos o más resistencias se conectan en paralelo entre dos terminales, A y B, la resistencia equivalente es siempre menor que la resistencia de valor más bajo. La lógica de este postulado se demostrará analizando la siguiente figura.

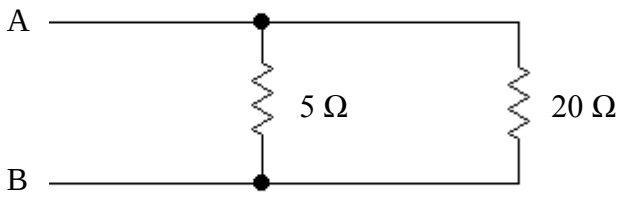


FIGURA 4.4. RESISTENCIAS EN PARELELO.

En este circuito, inicialmente se conecta una resistencia de 5ohms (R_1) entre las terminales A y B. Si se conecta otra resistencia de 20 ohms (R_2) en paralelo con la de 5 ohms (R_1), es evidente que la oposición al flujo de la corriente entre A y B será menor que antes. Esto se debe a que ahora, la corriente puede fluir por otra trayectoria que no existía cuando el

circuito contaba únicamente con la resistencia de 5 ohms (R_1). La resistencia equivalente de varias resistencias conectadas en paralelo se determina mediante la siguiente ecuación

$$\frac{1}{R_{equivalente}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} \cdots \quad (2)$$

Para el caso particular en que se tienen sólo dos resistencias en paralelo, la resistencia equivalente única se determina por medio de la ecuación

$$R_{equivalente} = \frac{(R_1 * R_2)}{(R_1 + R_2)}$$

Por lo tanto, la resistencia equivalente de 20 ohms en paralelo con 5 ohms es

$$\frac{(\mathbf{5} * \mathbf{20})}{(\mathbf{5} + \mathbf{20})} = \mathbf{4}\Omega$$

En consecuencia, se puede utilizar una sola resistencia de 4 ohms (R_3) para reemplazar las dos originales como en la siguiente figura.

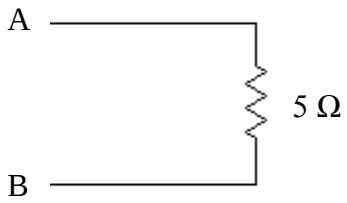


FIGURA 4.5. RESISTENCIA EQUIVALENTE.

4.5. LA LEY DE OHM

La resistencia eléctrica es la oposición que existe al flujo de la corriente en un circuito, y depende de muchos factores. El alambre de cobre, aunque se considera un buen conductor de corriente eléctrica, presenta cierta resistencia. El físico alemán, George Simon Ohm (1787-1854), descubrió que para un conductor metálico dado, de una longitud y corte

transversal específicos, la relación entre voltaje y la corriente era constante. Esta relación se conoce como resistencia y se expresa en la unidad ohm, denominada así en su honor.

La **Ley de Ohm** se considera a menudo como el fundamento del análisis de circuitos y se puede expresar mediante la fórmula:

$$R = \frac{E}{I} \tag{1}$$

En donde,

E= La diferencia de potencial entre los dos extremos de un elemento de resistencia (que se mide en volts.

I= La corriente eléctrica que pasa por dicho elemento de resistencia (que se mide en amperes, amp).

R= La resistencia del mismo elemento (que se mide en ohms).

Existen otras dos formas útiles que se pueden obtener de la ecuación (1), y son:

$$I = \frac{E}{R} \tag{2}$$

$$E = I * R \tag{3}$$

Para producir una corriente, primero debe de existir un voltaje en la resistencia. Los primeros experimentos en este campo, reconocieron el hecho de que una corriente eléctrica constituía un movimiento de cargas a lo largo de un conductor. El sentido del flujo de la corriente no se pudo determinar y, desgraciadamente, se convino en forma arbitraria que fuera desde un cuerpo de carga positiva hacia otro de carga negativa (positivo o negativo), y este acuerdo se estableció tan firmemente, que sigue en vigencia hasta nuestros días. Así, la dirección convencional o dirección positiva del flujo de la corriente, es siempre de positivo a negativo, aunque se sabe ahora que la dirección del flujo electrónico, que en realidad constituye una corriente eléctrica, va de negativo a positivo.

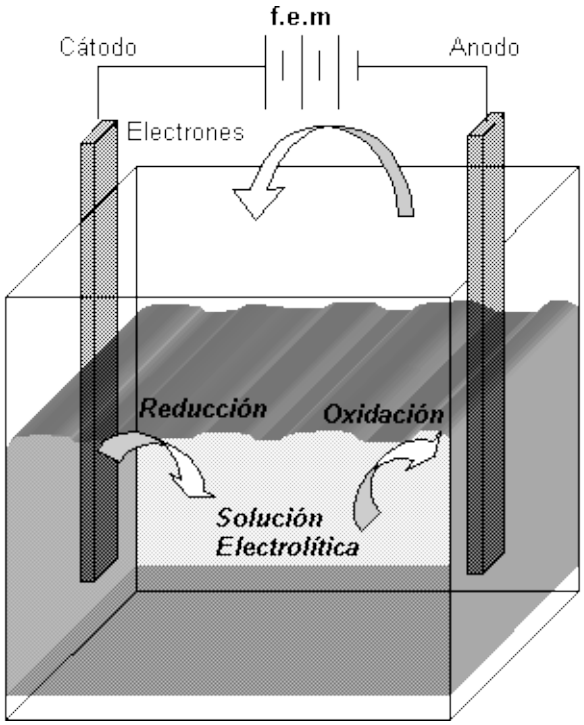


FIGURA 4.6. DIFERENCIA DE POTENCIAL (VOLTAJE).

El ***votl*** es una unidad de presión o el potencial eléctrico, y se mide con un voltímetro. Los voltímetros poseen una alta resistencia eléctrica y siempre se conectan en paralelo con un circuito o un componente.

El ***ampere*** es la unidad de la corriente eléctrica y se mide con un amperímetro. Los amperímetros tienen un baja resistencia interna y se conectan en serie con el circuito o el componente.

Los circuitos eléctricos en serie y en paralelo se pueden resolver aplicando la **Ley de Ohm** y conociendo las siguientes reglas:

a) En un **circuito en serie**, el **voltaje** de un grupo de resistencia es igual a la suma de los voltajes que pasan por cada una de ellas.
b) La **corriente** total que entra a un **circuito en paralelo**, es igual a la suma de las corrientes de cada rama en paralelo.
c) La **corriente** es igual en cada resistencia de un **circuito en serie**.
d) El **voltaje** es el mismo en todas las ramas de resistencia de un **circuito en paralelo**.

4.6. LEY DE KIRCHOFF

Frecuentemente, en los circuitos hay uniones o puntos comunes en donde se juntan varios cables o alambres. La característica interesante de estos puntos de cruce o unión consiste en

que la suma de todas las corrientes que salen del mismo. La razón de esto es que los electrones no se pueden acumular en dicho punto o unión, si no que debe de salir tan rápidamente como van llegando.

La unión que se ilustra en el diagrama esquemático de la figura 4.7 tiene cuatro cables que terminan en ella y en la misma figura se indican las corrientes que llevan.

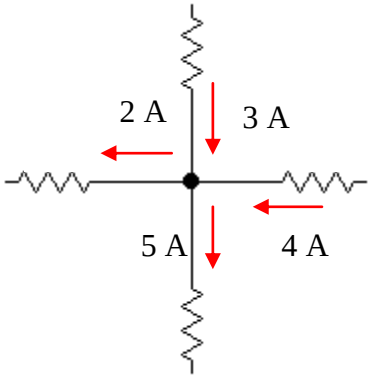


FIGURA 4.7. RESISTENCIA EQUIVALENTE.

La suma de las corrientes que llegan (3A+4A) es igual a la suma de las corrientes que salen (5A+2A), de manera que la ley se cumple. Esta ley se puede aprovechar, ya que permite calcular la corriente de cable o conductor sin necesidad de medirla.

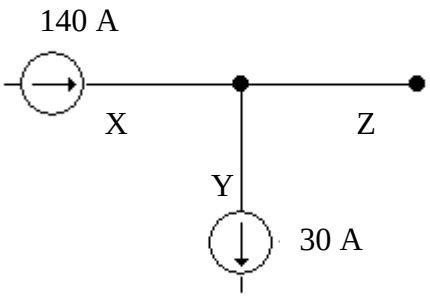


FIGURA 4.8. CALCULO DE LA CORRIENTE Z.

Los conductores X y Y llevan, respectivamente, 140 amps y 30 amps, con las direcciones indicadas. Puesto que hay 140 amps que llegan a la unión y sólo salen 30 por el alambre Y,

es evidente que el conductor Z debe portar los 110 amps restantes. Por lo tanto, se conoce el valor de la corriente del conductor Z sin haberla medido.

4.7. POTENCIA DE CIRCUITOS DE C-D (Corriente Directa).

El propósito de una fuente de energía (alimentación), en un circuito eléctrico, es suministrar la energía eléctrica necesaria a la carga que empleará dicha energía para efectuar una función útil o un trabajo. En electricidad, el trabajo se realiza mediante el movimiento de electrones (corriente eléctrica). La potencia es la velocidad con la que se hace un trabajo. Una fuerza electromotriz de un volt, que produce una corriente de un ampere (a través de una resistencia de un ohm), proporciona un watt de potencia.

La potencia eléctrica (watts) proporcionada a una carga es siempre igual al producto del voltaje en c-d de la carga por la corriente de c-d que pasa por ella, como lo indica a continuación.

$$P = E * I$$

en donde, P = potencia en watts
E = voltaje en volts
I = corriente en amperes

Si a un motor de c-d (corriente directa) se le suministra potencia eléctrica, parte de está se convertirá en energía mecánica y la restante se convertirá en calor. Cuando se suministra potencia a una batería o acumulador (mientras se carga), parte de la potencia se convierte en energía química y la restante en calor. No obstante, cuando se suministra cierta potencia a una resistencia, toda ella se convierte en calor. Esta conversión de energía eléctrica en energía térmica es, por lo tanto, un proceso muy eficiente y se aprovecha diariamente en aparatos tales como tostadores, estufas y calefacción etc. Al igual que se tiene tres formas de expresión de la ley de Ohm, también existen tres maneras de relacionarla potencia con el voltaje y la corriente, que son:

$$P = E * I$$

$$E = \frac{P}{I}$$

$$I = \frac{P}{E}$$

La potencia eléctrica de un circuito de c-d se puede encontrar utilizando la ecuación:

$$P = E * I$$

En donde P = potencia en watts
I = corriente en amperes
E = voltaje en volts

Puesto que el voltaje E y la corriente I están relacionadas por la resistencia R (de acuerdo con la ley de Ohm), se puede derivar dos nuevas expresiones de la ecuación (1). Cuando se substituye E por IR, la ecuación (1) se convierte en:

$$P = IR * I$$

$$P = I^2 * R$$

y, puesto que,

$$I = \frac{E}{R}$$

También se puede substituir I con E/R en la ecuación (1)

$$P = \frac{E}{R} * E$$

$$P = \frac{E^2}{R}$$

Por lo tanto, ahora se puede calcular la potencia de cualquier circuito de c-d, usando el término de R y E o I (no es necesario conocer el valor de ambos a la vez).

4.8. ELECTRÓNICA DIGITAL Y ANALÓGICA

De hecho, los términos **analógico** y **digital** no son ideales. *Analógico* viene de la palabra *Analogía*, que significa que la señal se parece a la magnitud física que representa; sin embargo, una señal discreta también podría ser una analogía de una magnitud cambiante.

De manera similar, digital implica el uso de números, aunque muchas señales discretas no constituyen la representación directa de los números. Podría representar más preciso usar los términos continuo y discreto, pero como los términos *analógico* y *digital* se usan de manera universal, son los que se utilizaran en el texto. Son muchas razones para representar magnitudes físicas mediante **señales eléctricas**. Resulta muy sencillo procesar las señales eléctricas mediante circuitos electrónicos que son tanto económicos como confiables. Las señales electrónicas pueden transmitirse sin dificultad a las largas distancias y también pueden almacenarse para reproducirlas mas tarde. Sin embargo, ¡no tendría sentido generar y procesar estas señales si al final no fueran a utilizarse para algo!

4.8.1. Sistemas Electrónicos.

En términos generales, un **sistema** es cualquier volumen cerrado del cual se conocen todas las entradas y salidas. Un **sistema electrónico** es cualquier disposición de componentes electrónicos con un conjunto definido de entradas y salidas.

4.8.2. Sistemas Digitales Eléctricos.

El uso de máquinas de todos tipos por parte del hombre ha originado una creciente dependencia de las técnicas digitales más que de las analógicas. Una razón de ello es que hay una gran cantidad de sensores y actuadores binarios que son más sencillos que los tipos analógicos correspondientes. Por ejemplo, es mucho mas fácil ENCENDER o APAGAR una lámpara mediante un conmutador que variar su potencia de manera continúa. Otra razón para el uso extensivo de las técnicas digitales es que las magnitudes digitales se pueden transmitir, procesar y almacenar con mayor facilidad que sus equivalentes analógicas, y resultan menos afectadas por la presencia de ruido.

El control digital, y en particular el binario, es común en todos los campos de la vida, desde los sistemas domésticos de calefacción, que encienden o apagan calentadores dependiendo de si la temperatura está por encima o por debajo de cierto valor, hasta complejos computadores de control de vuelos, que recogen información de diversos sensores y producen señales de salida para determinar el funcionamiento de un conjunto de actuadores.

Desde los años setenta del siglo XX, esta capacidad de procesamiento se ha hecho posible en la forma de los microprocesadores, que proporcionan las funciones de cálculo de un computador dentro de un solo circuito integrado. El bajo costo y la flexibilidad de estos componentes ha provocado su uso en miles de aplicaciones, desde las calculadoras hasta los automóviles y desde grabadoras de videos hasta cafeteras.

Los microprocesadores son circuitos electrónicos relativamente complejos que contienen entre 10^5 y 10^6 transistores. Sin embargo, los circuitos se basan en combinaciones de

elementos mucho más pequeños llamados compuertas, las cuales constituyen en sí circuitos sencillos que utilizan poca cantidad de transistores y componentes pasivos.

4.9 ESTADOS LÓGICOS

Los estados lógicos son aquellos dispositivos que pueden tomar tan sólo dos estados, como ejemplos podemos mencionar:

a) Un conmutador que sólo puede estar ENCENDIDO o APAGADO

b) Una válvula hidráulica que sólo puede estar ABIERTA o CERRADA, y

c) Un calentador eléctrico que sólo puede estar ENCENDIDO o APAGADO.

Tales valores se denominan **magnitudes binarias**, es común representar magnitudes semejantes mediante **variables binarias**, que son sólo nombres simbólicos para las magnitudes como se ilustra de un circuito binario en la siguiente figura.

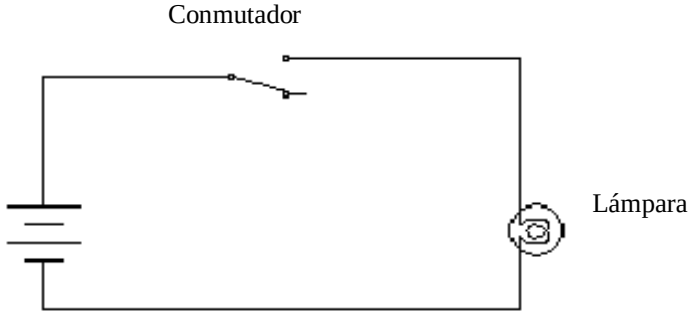


FIGURA 4.9. CIRCUITO BINARIO SENCILLO.

Si representamos el estado del conmutador mediante la variable binaria S y el estado de la lámpara mediante la variable binaria L, podemos representar de manera simbólica la relación entre las dos variables como

También podemos usar un nombre simbólico para el estado de cada variable, de modo que en vez de usar términos como ABIERTO y CERRADO o ENCENDIDO (ON) y APAGADO (OFF), podemos usar los símbolos “0” y “1”, ya que “0” representa el

conmutador ABIERTO y la lámpara APAGADA, cuya información se interpretará en la siguiente tabla.

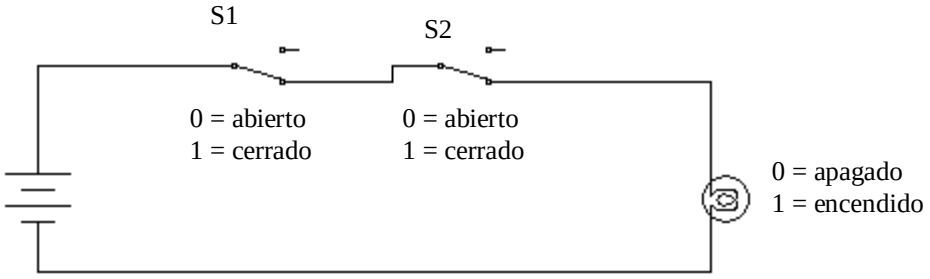
(Swich) S	L (Lámpara)
0	0
1	1

TABLA 4.1. TABLA DE VERDAD.

La tabla representa a la izquierda todos los estados posibles de conmutador e indica, a la derecha, los estados correspondientes de la lámpara. Esta tabla se llama **tabla de verdad** y define la relación entre las dos variables.

La siguiente figura 4.10 muestra un circuito que incorpora dos conmutadores en serie. Aquí es necesario que ambos conmutadores estén cerrados para que la lámpara se encienda. La relación entre las posiciones de los conmutadores y el estado de la lámpara se da en la tabla de verdad de la figura 4.10.

Esta relación **AND** es muy común en sistemas electrónicos y se encuentra en una gran variedad de aplicaciones cotidianas.



a) Circuito

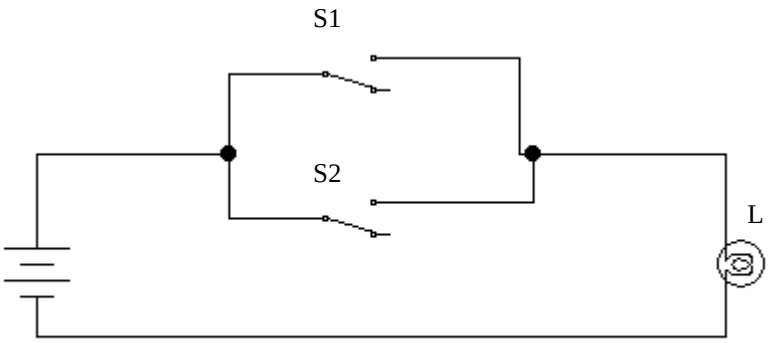
S1	S2	L
0	0	0
0	1	0
1	0	0
1	1	1

b) Tabla de verdad

FIGURA 4.10. DOS CONMUTADORES EN SERIE.

La figura 4.11. muestra un circuito que tiene dos conmutadores en paralelo. En esta configuración la lámpara se encenderá si cualquiera de los dos conmutadores se cierra. Esta función se describe en la tabla de verdad de la figura, en la cual los significados de “0” y “1” son iguales que los del ejemplo anterior. Como antes, podemos expresar esta relación

en palabras como "la lámpara se encenderá si, y sólo si, el conmutador S1 (Swich 1) esta cerrado O (OR) el conmutador S2 (Swich 2) está cerrado (o si ambos están cerrados)" o, en la forma abreviada.



a) Circuito

S1	S2	L
0	0	0
0	1	1
1	0	1
1	1	1

b) Tabla de verdad

FIGURA 4.11 .DOS CONMUTADORES EN PARALELO.

Un ejemplo de la función **OR,** de nuevo en una aplicación automotriz, podría ser la luz de la cortesía, que se enciende si se abre la puerta del conductor (cerrando un conmutador) O (OR) si se abre la puerta del pasajero.

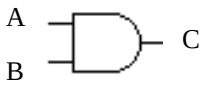
Los ejemplos de las funciones AND y OR se pueden extender al uso de tres o más conmutadores, como lo ilustra la figura, donde cualquier número de conmutadores se pueden conectar en serie o en paralelo.

4.10. COMPUERTAS LÓGICAS

Una compuerta lógica es un elemento que toma una o más señales binarias de entrada y produce una salida binaria apropiada, depende del estado o estados de las entradas. Hay tres tipos elementales de compuertas, dos de las cuales ya hemos visto, las funciones AND y OR.

4.10.1. Compuerta AND

La salida de una compuerta AND es verdadera (1) si, y sólo si, todas las entradas son verdaderas. La compuerta puede tener cualquier número de entradas. En la figura 4.12 aparece el símbolo lógico y la tabla de verdad para una compuerta AND de dos entradas. Las etiquetas de las entradas y salidas son arbitrarias.



A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

a) Símbolo de circuito. b) Tabla de verdad.

FIGURA 4.12. COMPUERTA AND DE DOS ESTRADAS.

4.10.2. Compuerta OR

La salida de un compuerta OR es verdadera (1) si y sólo si, por lo menos una de las entradas es verdadera. También se llama compuerta **OR inclusiva** por las razones mencionadas antes. La compuerta puede tener cualquier número de entradas. En la figura 4.13 aparecen el símbolo lógico y la tabla de verdad para una compuerta OR de dos entradas.

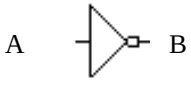
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

a) Símbolo de circuito. b) Tabla de verdad.

FIGURA 4.13. COMPUERTA OR DE DOS ESTRADAS.

4.10.3. Compuerta NOT

La salida de una compuerta NOT es verdadera (1) si y sólo si, su única entrada es falsa. Ésta compuerta tiene la función de un **INVERSOR lógico** pues la salida es el **complemento** de la entrada. En ocasiones esta compuerta se conoce como compuerta de INVERSIÓN o simplemente como INVERSOR. En la figura 4.14 aparecen el símbolo de circuito y la tabla de verdad para una compuerta NOT.



A	B
0	1
1	0

a) Símbolo de circuito. b) Tabla de verdad.

FIGURA 4.14. COMPUERTA NOT o INVERSOR.

El circuito en el símbolo de INVERSOR representa el proceso de inversión. El símbolo triangular sin el circuito representaría una función en la que el estado de la salida seria idéntico al de la entrada. Esta función recibe el nombre de ***buffer.*** En electrónica un ***buffer*** es un dispositivo que evita el efecto de carga en un circuito. En su forma más sencilla es un amplificador operacional funcionando como seguidor. Por consiguiente el voltaje y la corriente no disminuye en el circuito, ya que éste toma el voltaje de la fuente de alimentación del operacional y no de la señal que se está introduciendo, por lo que si una señal llegara con poca corriente, el circuito seguidor compensaría esa pérdida con la fuente de alimentación del amplificador operacional, ya sea éste unipolar o bipolar).

La presencia de un buffer no afecta el estado de la señal lógica. Sin embargo, cuando se llegarán a considerar la puesta en práctica de compuertas que usan circuitos electrónicos, veremos que se puede usar un buffer para cambiar las propiedades eléctricas de una señal lógica.

Resulta interesante hacer notar que el símbolo para un buffer es similar al que se usa para un amplificador analógico de una sola entrada. Como el buffer no produce función lógica alguna, por lo común no se considera como una compuerta elemental. Sin embargo, para efectos de brindar un concepto completo se proporciona su símbolo lógico y tabla de verdad en la siguiente figura.

A	B
0	0
1	1

a) Símbolo de circuito. b) Tabla de verdad.

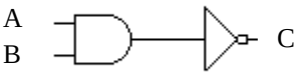
FIGURA 4.15. BUFFER LÓGICO.

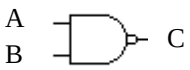
Compuertas compuestas

Las compuertas elementales recién descritas se pueden combinar para formar cualquier función lógica deseada.

4.10.4. Compuerta NAND

Una compuerta NAND puede tener cualquier número de entradas. En la figura aparece el circuito equivalente, el símbolo lógico y la tabla de verdad para una compuerta NAND de dos entradas.





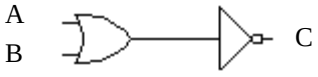
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

a) Circuito equivalente. b) Símbolo de circuito. c) Tabla de verdad.

FIGURA 4.16. COMPUERTA NAND DE DOS ENTRADAS.

4.10.5. Compuerta NOR

Una compuerta NOR puede tener cualquier número de entradas. La figura muestra el circuito equivalente, el símbolo lógico y la tabla de verdad para una compuerta NOR de dos entradas.



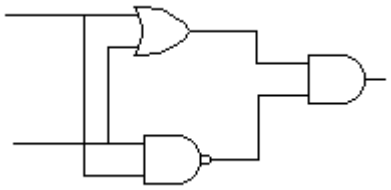
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

a) Circuito equivalente. b) Símbolo de circuito. c) Tabla de verdad.

FIGURA 4.17. COMPUERTA NOR DE DOS ENTRADAS.

4.10.6. Compuerta OR exclusiva

La salida de una compuerta OR exclusiva es verdadera (1) si, sólo si, una u otra de sus entradas es verdadera, pero no si ambas son verdaderas.



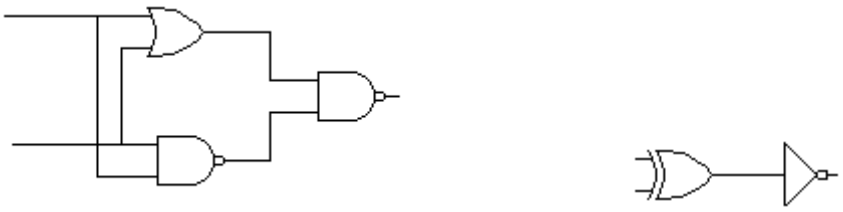
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

a) Circuito equivalente. b) Símbolo de circuito. c) Tabla de verdad.

FIGURA 4.18. COMPUERTA NOR DE DOS ENTRADAS.

4.10.7. Compuerta NOR exclusiva

El último miembro de nuestro grupo lógico de compuertas es la compuerta NOR exclusiva, la cual, como su nombre lo indica, es la navegación de la compuerta OR exclusiva. Se puede considerar como una compuerta OR exclusiva seguida de un INVERSOR o como el circuito equivalente de la figura, en el cual la compuerta AND es remplazada por una compuerta NAND. Esta compuerta da una salida verdadera cuando ambas entradas son 0 o cuando ambas son 1. Por lo tanto, proporciona una salida verdadera cuando las entradas son iguales. Es por ello que esta compuerta también se conoce como **compuerta de equivalencia** o **de igualdad**. La figura que esta a continuación, muestra los circuitos equivalentes, un símbolo lógico y una tabla de verdad para la compuerta NOR exclusiva.



a) Circuitos equivalentes.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

b) Símbolo de circuito. c) Tabla de verdad.

FIGURA 4.19. COMPUERTA NOR DE DOS ENTRADAS.

4.11. ÁLGEBRA BOOLEANA

El álgebra booleana recibe el nombre de su inventor, el matemático del siglo XIX George Boole (1854), quién formuló un conjunto básico de reglas con respecto a la lógica binaria, es decir, de falso-verdadero. De hecho, no fue hasta muchos años después de la muerte de Boole que se hizo evidente la importancia de su trabajo a través de los estudios del estadounidense, Claude Shannon (1938). En 1938 Shannon, que era el alumno de posgrado del Massachusetts Institute of Technology, aplicó el algebra booleana al diseño de redes telefónicas de conmutación. A partir de este trabajo se hizo evidente que aquella constituía una base para el análisis de todos los tipos de sistemas binarios.

El algebra booleana define constantes, variables y funciones para describir sistemas binarios. Luego describe cierto número de teoremas que se pueden usar para manipular expresiones lógicas.

4.11.1. Funciones Booleanas

Cada una de las funciones lógicas elementales está representada dentro del álgebra booleana mediante un símbolo único, como se muestra en la siguiente tabla.

Función	Símbolo	Ejemplo
AND	punto	C=A·B = AB
OR	más	C=A+B
NOT	barra	C=A

TABLA 4.1. SÍMBOLOS.

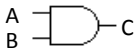
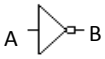
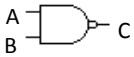
Función	Símbolo	Representación booleana	Tabla de verdad
AND		C = A * B	A B C 0 0 0 0 1 0 1 0 0 1 1 1
OR		C = A + B	A B C 0 0 0 0 1 1 1 0 1 1 1 1
NOT		B= A	A B 0 1 1 0
NAND		C= A * B	A B C 0 0 1 0 1 1 1 0 1 1 1 0
NOR		C = A + B	A B C 0 0 1 0 1 0 1 0 0 1 1 0
OR exclusiva		C = A B	A B C 0 0 0 0 1 1 1 0 1 1 1 0
NOR exclusiva		C = A B	A B C 0 0 1 0 1 0 1 0 0 1 1 1

TABLA 4.2. COMPUERTAS LÓGICAS.

Las expresiones booleanas no son únicas.

Una descripción más formal del proceso podrá ser la siguiente:

1ª Se genera un **minterm** por cada columna de la tabla de verdad en la cual aparezca un "1".

2ª El minterm contiene cada variable de entrada en orden; la entrada no está negada si es "1" y negada si es un "0".

3ª La expresión global para la función lógica es entonces la suma de los minterms (donde la salida se obtiene "1").

Este proceso se puede aplicar a tablas de verdad de cualquier tamaño; por ejemplo:

A	B	C	D		minterms
0	0	0	0		
0	0	1	1	$\bar{A}\bar{B}C$	
0	1	0	0		
0	1	1	0		
1	0	0	1	$A\bar{B}\bar{C}$	
1	0	1	0		
1	1	0	0		
1	1	1	1	ABC	

Corresponde a la expresión lógica:

$$D=\bar{A}\bar{B}C+A\bar{B}\bar{C}+ABC$$

4.11.2. Extracción de la expresión booleana de un sistema a partir de su diagrama lógico

Considérese el siguiente circuito lógico:

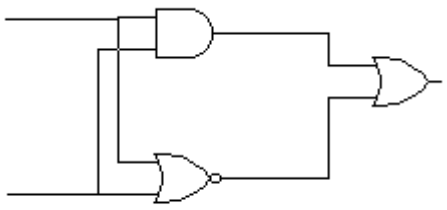


FIGURA 4.20. CIRCUITO LÓGICO.

El método más sencillo de extraer la expresión booleana para este circuito consistente tan sólo en escribir sobre el diagrama la salida de cada compuerta en relación con sus entradas, empezando con las compuertas cercanas las entradas y trabajando de ahí a la salida.

4.11.3. Generación de un diagrama lógico de un sistema a partir de su expresión booleana

Considérese la siguiente expresión booleana:

$$C = AB + \overline{\bar{A}\,\bar{B}} + (A+B)$$

El desarrollo del diagrama lógico es el inverso del proceso ilustrado en el ejemplo. Aquí empezamos en la salida y trabajamos hacia la entrada. El lado derecho de la expresión tiene tres componentes que están unidos por la función OR. Por lo tanto, la salida de nuestro circuito vendrá de una compuerta OR de tres entradas. Las entradas a esta compuerta serán los tres componentes de la expresión. La primera, $A\bar{B}$, proviene de una compuerta AND de dos entradas con entradas A y B, la segunda de una compuerta NAND de dos entradas con entradas A y B y la tercera de una compuerta OR de dos entradas también con entradas A y B. el diagrama lógico completo aparece a continuación.

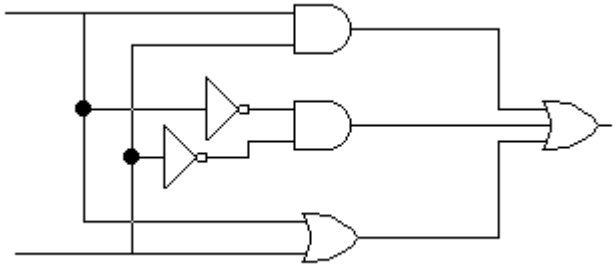


FIGURA 4.21. CIRCUITO LÓGICO.

4.11.4. Teoremas booleanos

Las reglas de álgebra booleana consisten en un conjunto de **identidades** y un conjunto de **leyes**.

Es importante entender el proceso de simplificación algebraica pues constituye una técnica útil para funciones sencillas. Sin embargo, para expresiones más complejas de ordinario se buscan métodos más complejos, uno de ellos son los mapas de Karnaugh.

4.12. Mapas de Karnaugh

El **mapa de Karnaugh** es un método gráfico de representación de la información que se encuentra en una tabla de verdad (Karnaugh, 1953).

En una tabla de verdad cada representación posible de las entradas esta representada mediante una línea única en la tabla, mientras que el valor de la salida que corresponde a ese patrón de entradas aparece en la columna de salida. En un mapa de Karnaugh cada combinación posible de entradas está representada por una caja dentro de una rejilla y el valor correspondiente de la salida se escribe dentro de esa caja. La figura 4.23 muestra un ejemplo de eso para una función de dos entradas, A y B.

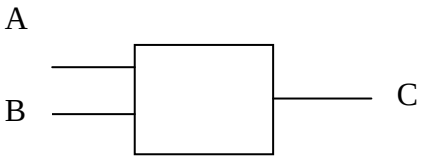
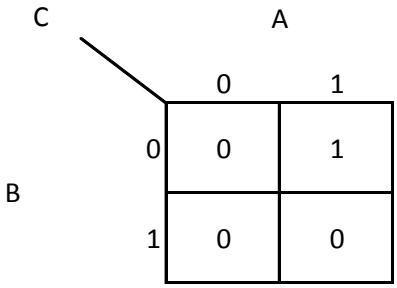


FIGURA 4.22. SISTEMAS DE DOS ESTRADAS Y UNA SALIDA.

A	B	C
0	0	0
0	1	0
1	0	1
1	1	0

(a) Tabla de verdad.



(b) Mapa de Karnaugh.

FIGURA 4.23. TABLA DE VERDAD Y MAPA DE KARNAUGH DE DOS ENTRADAS.

Este principio se mantiene conforme se agregan más cajas para representar sistemas con un mayor número de entradas. La figura 4.24 muestra mapas de Karnaugh para sistemas con tres y cuatro entradas. Esta técnica se puede extender a sistemas con un número mayor de entradas, si así se requiere.

Para un sistema con tres variables de entradas se utiliza un conjunto de cuatro por dos, como en la figura 4.24. Dos de las variables están relacionadas con las cuatro columnas y la variable restante con las dos filas. La distribución de las variables aparece arriba y a un lado del mapa. Los valores de las variables apropiadas se muestran junto a cada fila y columna. Entonces, la caja y corresponde a AB cuando es "01" y a C cuando es "0". La caja V es, por lo tanto, B. De manera similar, la caja W es $A\overline{BC}$. Los sistemas con cuatro entradas requieren una disposición de cuatro por cuatro, como se ve en la figura. Aquí la caja X corresponde a $\overline{A}BCD$, la caja Y a ABCD y la caja Z a $\overline{A}BC\overline{D}$. Notará el lector que la rejilla no esta marcada en orden binario. De hecho, la secuencia corresponde al **código de Gray**.

Observar que la secuencia es tal que los cuadros adyacentes, tanto en forma horizontal como vertical, difieren en el estado de una variable. Por ejemplo, X y Y difieren sólo en le estado de A, mientras que X y Z difieren sólo en el estado de D. La importancia de esta propiedad se hará evidente. Nótese que esta relación no sólo se aplica la caja que estén unidas en diagonal, como con Y y Z, que difiere en los estados tanto de A como de D.

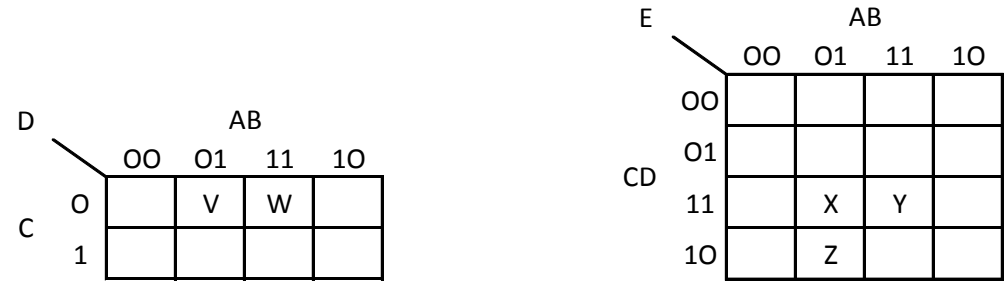


FIGURA 4.24. MAPAS DE KARNAUGH PARA UN SISTEMA DE TRES Y CUATRO ENTRADAS.

En la figura 4.25 se muestran dos mapas de Karnaugh para las funciones E y F. Donde se puede extraer una expresión algebraica para las funciones directamente de los mapas de manera similar a la que se usó en el ejemplo para extraer esta información de una tabla de verdad. Es evidente que E está dada por la expresión.

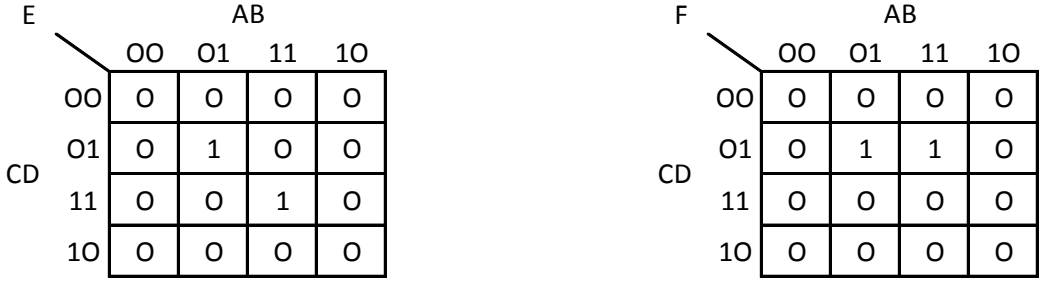


FIGURA 4.25. MAPAS DE KARNAUGH BÁSICOS.

$$E = \bar{A}\bar{B}CD + A\bar{B}CD$$

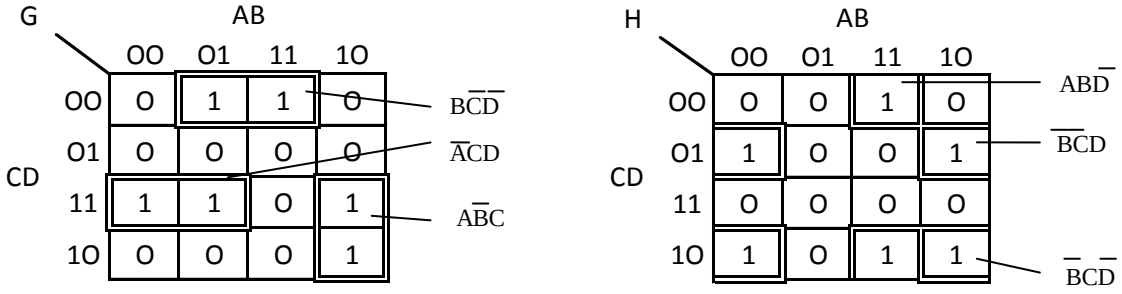
y que F está dada por

$$F = \bar{A}\bar{B}CD + A\bar{B}\bar{C}D$$

En este punto es interesante tratar de simplificar estas expresiones en forma algebraica utilizando las técnicas descritas antes. Si se hace esto, se descubrirá que E no se puede simplificar, ya que están en diagonal los 1`s.

La figura 4.26 muestra algunos ejemplos de las combinaciones de dos elementos. Nótese que para el propósito de combinar elementos que son “1”, las filas superior e inferior se consideran adyacentes, al igual que las columnas de derecha a izquierda. Es como si el

mapa fuera cilíndrico en cada plano de tal modo que la parte superior y la parte inferior y los dos lados se tocaran.



$G = B\bar{C}\bar{D} + \bar{A}CD + A\bar{B}D$ (a)

$H = AB\bar{D} + \bar{B}\bar{C}D + \bar{B}C\bar{D}$ (b)

FIGURA 4.26. MAPAS DE KARNAUGH SENCILLOS.

Considérese el mapa de la figura 4.27, aquí los cuatro "1`s" están colocados en un cuadrado. El mapa se puede describir mediante la expresión

$$E = A\bar{B}C\bar{D} + ABC\bar{D} + A\bar{B}CD + ABCD$$

Esto se puede simplificar combinando los términos 1º, y 2º, y 3º y 4º.

$$E = B\bar{C}D(A+\bar{A}) + BCD(A+\bar{A})$$

$$= B\bar{C}D + BCD$$

Esta simplificación es evidente a partir del mapa en el sentido de que podríamos dibujar lazos alrededor de estos dos pares de elementos. Sin embargo, los dos términos resultantes se podrían combinar entre sí:

$$= BD(C+C)$$

$$= BD$$

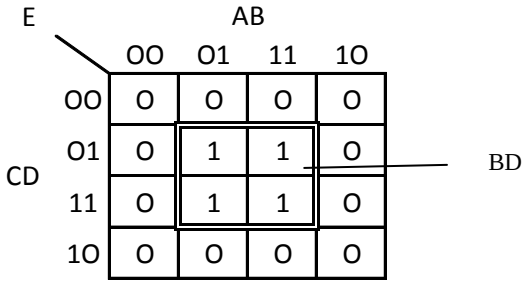


FIGURA 4.27. MAPAS DE KARNAUGH CON UN CONJUNTO DE CUATRO "1".

CAPÍTULO 5

APLICACIÓN EN PUERTOS DE LA PC

5.1. SISTEMA OPERATIVO

El sistema operativo (SO) es un software de sistema, es decir, un conjunto de programas de computadora destinado a permitir una administración eficaz de sus recursos. Comienza a trabajar cuando se enciende el computador, y gestiona el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que utilicen microprocesadores para funcionar, ya que gracias a éstos podemos entender la máquina y que ésta cumpla con sus funciones (teléfonos móviles, reproductores de DVD, autoradios, computadoras, etc.).

Un sistema operativo puede ser contemplado como una colección organizada de extensiones software del hardware, consistente en rutinas de control que hacen funcionar un computador y proporcionan un entorno para la ejecución de los programas. Otros programas se apoyan en las facilidades proporcionadas por el sistema operativo para obtener acceso a los recursos del sistema informático, tales como archivos y dispositivos de entrada/salida (E/S). Los programas invocan generalmente por medio de ***llamadas al sistema operativo***. Además, los usuarios pueden interactuar con el sistema operativo directamente por medio de ***órdenes del sistema operativo***. En cualquier caso, el sistema operativo actúa como interfaz entre los usuarios y el hardware de un sistema informático.

El rango y la extensión de los servicios proporcionados por un sistema operativo dependen de varios factores. Entre otras cosas, las funciones visibles al usuario de un sistema operativo están en gran medida determinadas por las necesidades y características del entorno objetivo que el SO está destinado a soportar. Por ejemplo, un sistema destinado al desarrollo de programas en un entorno interactivo puede tener un conjunto bastante diferente de llamadas y órdenes que el de tiempo real dedicado, tal como el control del motor de un coche.

Internamente, un sistema operativo actúa como gestor de los recursos del sistema informático, tales como el procesador, la memoria, los archivos y los dispositivos de E/S. En esta función, el sistema operativo lleva la cuenta del estado de cada recurso y decide quién obtiene un recurso, durante cuánto tiempo y cuándo. En sistemas que soportan ejecución concurrente de programas, el sistema operativo resuelve las peticiones conflictivas de recursos de manera que preserva la integridad del sistema, y al hacerlo intenta optimizar el rendimiento final.

5.1.1. FUNCIONES PRINCIPALES.

Un sistema operativo desempeña cinco (5) funciones principales en la operación de un sistema informático: suministro de interfaz al usuario, administración de recursos, administración de archivos, administración de tareas y usuarios, servicio de soporte y utilidades.

5.1.1.1. Interfaces del usuario

Es la parte del sistema operativo que permite comunicarse con el de tal manera que se puedan cargar programas, acceder archivos y realizar otras tareas. Existen tres tipos básicos de interfaces, las cuales son: las que se basan en comandos, las que utilizan menús y las interfaces gráficas de usuario.

5.1.1.2. Gestión o administración de recursos

El manejo de recursos se clasifican en dos principalmente: **El Centralizado** que permite utilizar los recursos de una sola computadora, y **El Distribuido** que utiliza los recursos de más de una computadora al mismo tiempo.

Y en cualquiera de los dos casos su función es administrar, gestionar los recursos de hardware y de redes de un sistema informativo, por ejemplo ***el CPU, procesos o programas en ejecución, la memoria principal, los dispositivos de almacenamiento secundario y periféricos de entrada y de salida (E/S).***

5.1.1.2.1. Gestión de procesos

Un proceso es simplemente, un programa en ejecución que necesita recursos para realizar su tarea: tiempo de CPU, memoria, archivos y dispositivos de Entrada y Salida (E/S). Para ello el SO es el responsable de:

- Crear y destruir los procesos.
- Parar y reanudar los procesos.
- Ofrecer mecanismos para que se comuniquen y sincronicen.

La gestión de procesos podría ser similar al trabajo de oficina. Se puede tener una lista de tareas a realizar y a estas fijarles prioridades (alta, media, baja por ejemplo). Para ello se debe comenzar primero haciendo las tareas de prioridad alta y cuando se terminen seguir con las de prioridad media y después las de baja. Una vez realizada la tarea se marca. Esto puede traer un problema que las tareas de baja prioridad pueden que nunca lleguen a

ejecutarse. y permanezcan en la lista para siempre. Para solucionar esto, se puede asignar alta prioridad a las tareas más antiguas.

5.1.1.2.2. Gestión de la memoria principal

La memoria es una gran tabla de palabras o Bytes que se referencian cada una mediante una dirección única. Este almacén de datos de rápido accesos es compartido por la CPU y los dispositivos de E/S, es volátil y pierde su contenido en las fallas del sistema. El SO es el responsable de:

- Conocer qué partes de la memoria están utilizadas y por quién.
- Decidir qué procesos se cargarán en memoria cuando haya espacio disponible.
- Asignar y reclamar espacio de memoria cuando sea necesario.

5.1.1.2.3. Gestión del almacenamiento secundario

Un sistema de almacenamiento secundario es necesario, ya que la memoria principal (almacenamiento primario) es volátil y además muy pequeña para almacenar todos los programas y datos. También es necesario mantener los datos que no convenga almacenar en la memoria principal. Por lo que el SO se encarga de:

- Planificar los discos.
- Gestionar el espacio libre.
- Asignar el almacenamiento.

5.1.1.2.4. El sistema de Entrada y Salida (E/S)

Consiste en un sistema de almacenamiento temporal (caché), una interfaz de manejadores de dispositivos y otra para dispositivos específicos. El sistema operativo debe gestionar el almacenamiento temporal de E/S y realizar las interrupciones de los dispositivos de E/S.

5.1.1.3. Administración de archivos

Todo sistema de información contiene programas que administran los ***archivos***. Y los archivos son colecciones de información relacionada, definidas por sus creadores. Éstos almacenan programas (en código fuente, objeto y ejecutable) y datos tales como imágenes, textos, información de bases de datos, etc. Por lo que el SO es responsable de:

- Construir y eliminar archivos y directorios.
- Ofrecer funciones para manipular archivos y directorios.
- Establecer la correspondencia entre archivos y unidades de almacenamiento.
- Realizar copias de seguridad de archivos.

Organizar los diferentes Sistemas de Archivos de información que se almacena en las memorias (normalmente discos magnéticos) y en otros dispositivos de almacenamiento

secundarios de los computadores. Tales como: FAT, FAT32, EXT2, NTFS, que entre otras cosas permiten mantener el registro de la ubicación física de los archivos.

Desde el punto de vista del usuario estas diferencias pueden parecer insignificantes a primera vista, sin embargo, existen diferencias muy importantes. Por ejemplo, los sistemas de ficheros FAT32 y NTFS, que se utilizan fundamentalmente en sistemas operativos de Microsoft, tienen una gran diferencia para un usuario que utilice una base de datos con bastante información ya que el tamaño máximo de un fichero con un Sistema de Archivos FAT32 está limitado a 4 gigabytes sin embargo en un sistema NTFS el tamaño es considerablemente mayor.

5.1.1.4. Administración de tareas y de usuarios

Los programas de **administración de tareas** de un sistema operativo administran la realización de las tareas informáticas de los usuarios finales. Los programas controlan que áreas tiene acceso al CPU y por cuánto tiempo. Las funciones de administración de tareas pueden distribuir una parte específica del tiempo del CPU para una tarea en particular, e interrumpir al procesador en cualquier momento para sustituirla con una tarea de mayor prioridad. Y se clasifican en dos:

- **Monotarea:** Solamente puede ejecutar un proceso (aparte de los procesos del propio S.O.) en un momento dado. Una vez que empieza a ejecutar un proceso, continuará haciéndolo hasta su finalización y/o interrupción.
- **Multitarea:** Es capaz de ejecutar varios procesos al mismo tiempo. Este tipo de S.O. normalmente asigna los recursos disponibles (CPU, memoria, periféricos) de forma alternada a los procesos que los solicitan, de manera que el usuario percibe que todos funcionan a la vez, de forma concurrente.

En la **administración de usuarios** distribuye una parte específica del tiempo del CPU para el usuario o usuarios en particular, teniendo dos clasificaciones

- **Monousuario:** Sólo permite ejecutar los programas de un usuario al mismo tiempo.
- **Multiusuario:** Permite que varios usuarios ejecuten simultáneamente sus programas, accediendo a la vez a los recursos de la computadora. Normalmente estos sistemas operativos utilizan métodos de protección de datos, de manera que un programa no pueda usar o cambiar los datos de otro usuario.

5.1.1.5. Servicio de soporte

Los servicios de soporte de cada sistema operativo dependerán de la implementación particular de éste con la que se este trabajando. Entre las más conocidas se pueden destacar las implementaciones de Unix, desarrolladas por diferentes empresas de software; los sistemas operativos de Apple Inc., como Mac OS X para las computadoras de Apple; los

sistemas operativos de Microsoft, y las implementaciones de software libre, como Linux o BSD producidas por empresas, universidades, administraciones públicas, organizaciones sin fines de lucro y/o comunidades de desarrollo.
Estos servicios de soporte suelen consistir en:

- Actualización de versiones.
- Mejoras de seguridad.
- Inclusión de alguna nueva utilidad (un nuevo entorno gráfico, un asistente para administrar alguna determinada función).
- Controladores para manejar nuevos periféricos (este servicio debe coordinarse a veces con el fabricante del hardware).
- Corrección de errores de software.

No todas las utilidades de administración o servicios forman parte del sistema operativo, existen otros tipos importantes de software de administración de sistemas, como los sistemas de administración de base de datos o los programas de administración de redes. El soporte de estos productos es proporcionado por el fabricante correspondiente (que no tiene porque ser el mismo que el del sistema operativo).

5.1.2. LLAMADAS AL SISTEMA OPERATIVO (SO)

Los programadores de aplicaciones y sistemas suelen invocar servicios del sistema operativo desde sus programas por medio de llamadas al sistema, lo que a veces se denomina ***interfaces de programación de aplicaciones*** (API, application-programning interfaces). Las órdenes emitidas por los usuarios de lenguaje de órdenes suelen ser traducidas y se ejecutan como una serie de llamadas al sistema.

Además de proporcionar la mayor parte de la funcionalidad disponible a los usuarios del lenguaje de órdenes, las llamadas al sistema permiten generalmente un control más detallado sobre las operaciones del sistema y un acceso más directo a las facilidades hardware, especialmente al sistema de entrada/salida.

Excepto por algunas de las operaciones de apertura de sesión y de las funciones de gestión del sistema, las llamadas al sistema representan generalmente un superconjunto de las funciones disponibles a nivel de órdenes. Puesto muchas de ellas las hemos descrito en la sección anterior, discutiremos ahora algunas de las funciones típicamente proporcionadas llamadas al sistema.

Las llamadas al sistema para ejecución y control de programas incluyen generalmente un conjunto completo de los servicios disponibles mediante el lenguaje de órdenes, tales como EJECUTAR, ABORTAR y planificar con respecto al tiempo. Además los usuarios del llamado al sistema puede SUSPENDER uno o más programas hasta que ocurra una condición específica, REANUDAR programas previamente suspendidos y definir o

modificar varios atributos de tiempo de ejecución de programas. Suelen existir también algunas facilidades, con frecuencia extensas en sistemas de tiempo real, para comunicación y sincronización entre programas. Por ejemplo, los programas pueden intercambiar datos y señales de sincronización para sincronizar sus ejecuciones con ciertos sucesos.

Las llamadas al sistema para gestión de recursos proporcionan servicios para asignación, reserva y reclamación de los recursos del sistema. Por ejemplo, existen llamadas al sistema para extender o reducir la cantidad de memoria proporcionada al programa que lo invoca. Otros tipos de objetos pueden ser asignados o reservados por el programa invocante y destruidos o devueltos para su custodia por parte del sistema operativo.

Aunque son tratadas separadamente en algunos sistemas, las llamadas al sistema para gestión de dispositivos y archivos están funcionalmente combinadas en muchos sistemas operativos actuales. Básicamente, está clase de llamadas la sistema proporcionan facilidades para comunicarse con dispositivos de entrada/salida. Además de la lectura y escritura de los elementos individuales o de bloques de datos, se proporcionan llamadas tales como ABRIR y CERRAR para gestionar las conexiones lógicas con los dispositivos. Aunque virtualmente todos los dispositivos de E/S pueden ser accedidos en serie, se proporcionan llamadas especiales para acceso aleatorio o dispositivos con estructura de bloques, tales como discos. También se disponen de llamadas al sistema para inicialización de dispositivos y para la selección de modos específicos de operación.

5.1.2.3. Bibliotecas de interfaz de llamadas al sistema

Las llamadas al sistema no siempre tienen una expresión sencilla en los lenguajes de alto nivel, por ello se crean las bibliotecas de interfaz, que son bibliotecas de funciones que pueden usarse para efectuar llamadas al sistema. Las hay para distintos lenguajes de programación.

La aplicación llama a una función de la biblioteca de interfaz (mediante una llamada normal) y esa función es la que realmente hace la llamada al sistema.

5.1.3. INTERRUPCIONES Y EXCEPCIONES

El Sistema Operativo ocupa una posición intermedia entre los programas de aplicación y el hardware. No se limita a utilizar el hardware a petición de las aplicaciones ya que hay situaciones en las que es el hardware el que necesita que se ejecute el código del Sistema Operativo. En tales situaciones el hardware debe poder llamar al sistema, pudiendo deberse estas llamadas a dos condiciones:

a) Algún dispositivo de E/S necesita atención.

b) Se ha producido una situación de error al intentar ejecutar una instrucción del programa (normalmente de la aplicación).

En ambos casos, la acción realizada no está ordenada por el programa de aplicación, es decir, no figura en el programa.

Según los dos casos anteriores tenemos las interrupciones y las excepciones: donde la primera que es la **interrupción** señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que se estaba ocupando, ya ha terminado; y la segunda que es la **excepción** una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del SO.

5.1.3.1. Tratamiento de las interrupciones

Una interrupción se trata en todo caso, después de terminar la ejecución de la instrucción en curso.

El tratamiento depende de cuál sea el dispositivo de E/S que ha causado la interrupción, ante la cual debe poder identificar el dispositivo que la ha causado.

5.1.3.2. Importancia de las interrupciones

El mecanismo de tratamiento de las interrupciones permite al SO utilizar la CPU en servicio de una aplicación, mientras otra permanece a la espera de que concluya una operación en un dispositivo de E/S.

El hardware se encarga de avisar al SO cuando el dispositivo de E/S ha terminado y el SO puede intervenir entonces, si es conveniente, para hacer que el programa que estaba esperando por el dispositivo, se continúe ejecutando.

En ciertos intervalos de tiempo puede convenir no aceptar señales de interrupción, por ello las interrupciones pueden inhibirse por programa.

5.1.3.3. Excepciones

Cuando la CPU intenta ejecutar una instrucción incorrectamente construida, la unidad de control lanza una excepción para permitir al SO ejecutar el tratamiento adecuado. Al contrario que en una interrupción, la instrucción en curso es abortada. Las excepciones al igual que las interrupciones deben estar identificadas.

5.1.3.4. Clases de excepciones

Las instrucciones de un programa pueden estar mal construidas por las siguientes razones:

- El código de operación puede ser incorrecto.
- Se intenta realizar alguna operación no definida, como dividir por cero.
- La instrucción puede no estar permitida en el modo de ejecución actual.
- La dirección de algún operando puede ser incorrecta o se intenta violar alguno de sus permisos de uso.

5.1.3.5. Importancia de las excepciones

El mecanismo de tratamiento de las excepciones es esencial para impedir, junto a los modos de ejecución de la CPU y los mecanismos de protección de la memoria, que las aplicaciones realicen operaciones que no les están permitidas. En cualquier caso, el tratamiento específico de una excepción lo realiza el SO.

Como en el caso de las interrupciones, el hardware se limita a dejar el control al SO, y éste es el que trata la situación como lo convenga adecuado.

Es bastante frecuente que el tratamiento de una excepción no retorne al programa que se estaba ejecutando cuando se produjo la excepción, sino que el SO aborte la ejecución de ese programa. Este factor depende de la experiencia del programador para controlar la excepción adecuadamente.

5.2. SISTEMAS DE CONTROL

El término ***"Sistemas de Control"*** comprende dos términos que son sistema y control. Los cuales se pueden definir en forma individual:

a) **Un sistema** es una combinación de un número de componentes que están interconectados y que funcionan como una unidad, para poder lograr una meta específica.

b) **Control** es la habilidad de mantener el orden sobre un conjunto de variables.

Por lo que el ***"Sistema de Control"*** se puede definir como un grupo de componentes, los cuales pueden funcionar en forma conjunta para controlar diferentes variables, que rigen el comportamiento de un sistema. El cuerpo humano tiene sistemas, que controlan la temperatura del cuerpo. En una auditorio, puede encontrar un sistema que controle el aire

acondicionado. Los automóviles tienen sistemas que controlan el consumo de combustible y muchas otras cosas más.

Un sistema de control de invernadero, es un grupo de componentes que funcionan conjuntamente para regular dispositivos, los cuales mantienen la temperatura del aire, la humedad del suelo o la cantidad de luz ambiente en el invernadero.

Un sistema de control de velocidad mantiene una velocidad constante del motor, sin importar la carga cambiante en el motor.

Un sistema de control de luz puede mantener un nivel constante de luz, sin importar la cantidad de luz solar disponible. Las lámparas se ENCENDERÁN o se APAGARÁN, cuando no haya suficiente luz natural, de acuerdo al nivel de luz requerido.

Un sistema de control de nivel de agua mantiene un nivel fijo de líquido en un tanque o en un flujo continuo de agua en una tubería.

Algunas veces se requiere un sistema de control, el cual permite cambiar las condiciones de operación preestablecidas a menudo o mientras el sistema está en funcionamiento. Por ejemplo, se necesitan cambiar las condiciones de operación de un sistema de semáforos, conforme cambia la carga de tráfico, cambiar también los tiempos del ciclo de la lavadora de acuerdo a los diferentes ciclos de lavado, o cambiar las condiciones de operación de un invernadero en relación a la estación o las diferentes cosechas.

5.3. SISTEMA DE CONTROL COMPUTARIZADO

Un **Sistema de Control Computarizado** permite utilizar una computadora y un software de programación para supervisar el funcionamiento de un sistema de control. Estas operaciones se pueden cambiar fácilmente y a bajo costo realizando modificaciones en el software, sin hacer ninguna modificación complicada al sistema de circuitos o al dispositivo.

La computadora sirve como un dispositivo de control relativamente confiable y de bajo costo. Su ventaja más importante es la facilidad con la que se puede cambiar las condiciones de operación del sistema de control.

No es necesario realizar cambió en el sistema eléctrico de circuitos. No es necesario diseñar nuevos circuitos electrónicos. Todo lo que se requiere es, adaptar el programa de la computadora a las nuevas condiciones de operación y ejecutar el sistema.

Las computadoras, que generalmente se utilizan para el procesamiento de datos, se componen, como vimos en capítulos anteriores de cuatro unidades básicas:

Unidad de Entrada.- Tal como un teclado o un drive para disco, los cuales se utilizaron para alimentar información a la computadora.
CPU (Unidad Central Procesadora).- El "CEREBRO" de la computadora el cual recibe, descifra y realiza las instrucciones del programa y procesa los datos.

La memoria RAM.- La cual almacena los programas necesarios para realizar las tareas.

Unidad de Salida.- Tal como una pantalla, impresora o drive para disco, a través de los cuales la computadora "se comunica" con el mundo externo. Toma nota de que el drive realiza las funciones de entrada y salida.

La computadora de un controlador tiene todas estas unidades, pero también debe estar equipada con unidades (interfaz) que le permitan conectarse a dispositivos electromecánicos.

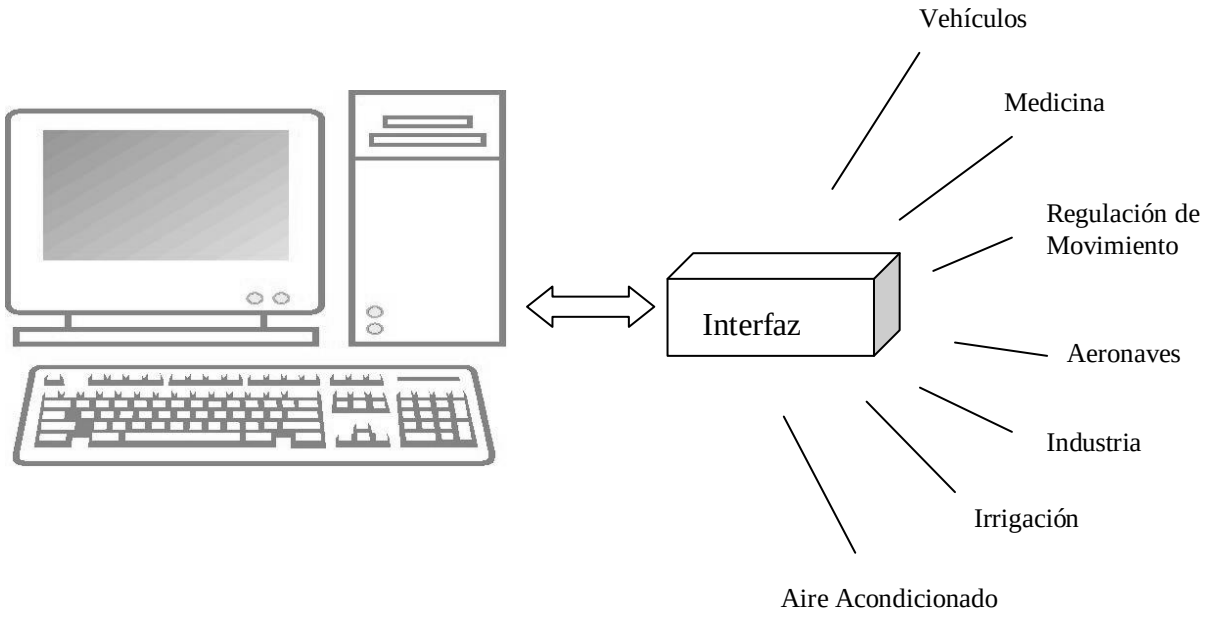


FIGURA 5.1. SISTEMA DE CONTROL COMPUTARIZADO.

Si se requiere conectar varios componentes y aparatos a una computadora de control, por lo que los dispositivos de entrada pueden ser: interruptores o aparatos de medición; mientras que los dispositivos de salida serán los: motores, luces, solenoides, etc. se pueden operar a través de la computadora.

Los dispositivos de entrada y de salida no se pueden conectar directamente al CPU de la computadora. Se necesita un tipo de dispositivo como intermediario. Un circuito electrónico, que sirva como árbitro entre el CPU y los distintos dispositivos de entrada y

salida, llamado INTERFAZ. Podemos visualizar una interfaz como una caja negra, la cual está conectada a varios dispositivos en un extremo y a la computadora en el otro extremo.

Algunas unidades pueden ser bidireccionales, es decir de entrada/salida. Pueden alimentar datos a la computadora o extraerlos de ésta.

Para llevar a cabo todo este control computarizado se requiere de **sensores** para percibir magnitudes externas y permitir la entrada de información a la computara y **actuadores** para controlar dichas magnitudes controlado por medio de la salida correspondiente del computador de manera eficaz. Los sensores y actuadores a menudo reciben el nombre de **transductores**.

5.4. SENSORES

Un sensor es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser por ejemplo: temperatura, intensidad luminosa, distancia, aceleración, inclinación, desplazamiento, la presión, la fuerza, torsión, humedad, acides (pH), etc. Una magnitud eléctrica obtenida puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una tensión eléctrica (como en un termopar), una corriente eléctrica (como un fototransistor), etc.

Un sensor se diferencia de un transductor, en que el primero está siempre en contacto con la variable a medir o a controlar. Recordando que la señal que nos entrega el sensor no solo sirve para medir la variable, sino también para convertirla mediante circuitos electrónicos en una señal estándar (4 a 20 mA, o 1 a 5 V dc) para tener una relación lineal con los cambios de la variable censada dentro de un rango, para fines de control de dicha variable en un proceso.

Puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. Como por ejemplo el termómetro de mercurio que aprovecha la propiedad que posee el mercurio de dilatarse o contraerse por la acción de la temperatura. Un sensor también puede decirse que es un dispositivo que convierte una forma de energía en otra.

Un sensor es un tipo de transductor que transforma la magnitud que se quiere medir o controlar en otra, que facilita su medida. Pueden ser de indicación directa (un termómetro de mercurio) o pueden estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, un computador y un display) de modo que los valores detectados puedan ser leídos por un humano.

Por lo general, la señal de salida de estos sensores no es apta para su lectura directa y a veces tampoco para su procesado, por lo que se usa un circuito de acondicionamiento,

como por ejemplo el puente de Wheatstone, amplificadores y filtros electrónicos que adaptan la señal a los niveles apropiados para el resto de la circuitería.

5.4.1. TEMPERATURA

La temperatura es una magnitud referida a las nociones comunes de calor o frío, por lo general un objeto más "caliente" tendrá una temperatura mayor. Físicamente es una magnitud escalar dada por una función creciente del grado de agitación de las partículas de los materiales. A mayor agitación, mayor temperatura. Así, en la escala microscópica, la temperatura se define como el promedio de la energía de los movimientos de una partícula individual por grado de libertad. En el caso de un sólido, los movimientos en cuestión resultan ser las vibraciones de las partículas en sus sitios dentro del sólido.

5.4.1.1. Termistor

Un termistor es un semiconductor que varía el valor de su resistencia eléctrica en función de la temperatura.

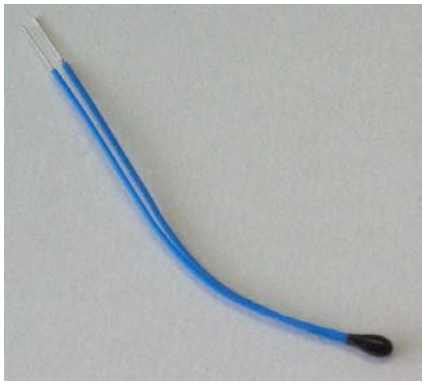


FIGURA 5.2. TERMISTOR.

Donde existen dos clases de termistores: NTC y PTC.

Termistor NTC (Negative Temperature Coefficient:) es una resistencia variable cuyo valor va decreciendo a medida que aumenta la temperatura. Son resistencias de coeficiente de temperatura negativo, constituidas por un cuerpo semiconductor cuyo coeficiente de temperatura es elevado, ya que, su conductividad crece muy rápidamente con la temperatura.

Se emplean en su fabricación óxidos semiconductores de níquel, zinc, cobalto, principalmente.

Termistor PTC (Positive Temperature Coefficient:) es una resistencia variable cuyo valor se ve aumentado a medida que aumenta la temperatura.

Estos termistores PTC se utilizan en una gran variedad de aplicaciones tales como: limitación de corriente, sensor de temperatura, desmagnetización y para la protección contra el recalentamiento de equipos tales como motores eléctricos. También se utilizan en indicadores de nivel, para provocar retardos en circuitos, como termostatos, y como resistores de compensación.

El termistor PTC pierde sus propiedades y puede comportarse eventualmente de una forma similar al termistor NTC si la temperatura llega a ser demasiado alta.

Las aplicaciones de un termistor PTC están, por lo tanto, restringidas a un determinado margen de temperaturas.

El **termostato** es el componente de un sistema de control que abre o cierra un circuito eléctrico en función de la temperatura.

Su versión más simple consiste en una lámina bimetálica. Como la que utilizan los equipos de aire acondicionado para apagar o encender el compresor.

Otro ejemplo lo podemos encontrar en los motores de combustión interna donde controlan el flujo del líquido refrigerante que regresa al radiador dependiendo de la temperatura del motor.

FIGURA 5.3. TERMOSTATO.

5.4.2. LUZ

La luz (*del latín lux, lucis*) es la clase de energía electromagnética radiante capaz de ser percibida por el ojo humano. En un sentido más amplio, el término luz incluye el rango

entero de radiación conocido como el espectro electromagnético. La ciencia que estudia las principales formas de producir luz, así como su control y aplicaciones se denomina óptica.

5.4.2.1. Panel fotovoltaico

Los módulos fotovoltaicos o colectores solares fotovoltaicos (llamados a veces **paneles solares**, aunque esta denominación abarca otros dispositivos), están formados por un conjunto de celdas (células fotovoltaicas) que producen electricidad a partir de la luz solar que incide sobre ellos. La potencia máxima que puede suministrar un módulo se denomina potencia pico.

Las placas fotovoltaicas se dividen en:

- Monocristalinas: se componen de secciones de un único cristal de silicio (reconocibles por su forma circular o hexagonal).
- Policristalinas: cuando están formadas por pequeñas partículas cristalizadas.
- Amorfas: cuando el silicio no se ha cristalizado.
- Su efectividad se incrementa cuanto mayor son los cristales, pero también aumenta su peso, grosor y coste.

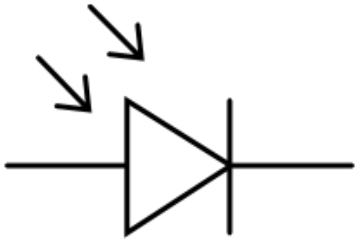
El rendimiento de las primeras puede alcanzar el 20% mientras que el de las últimas puede no llegar al 1%, sin embargo su coste y peso es muy inferior.

FIGURA 5.4. PANELES SOLARES.

5.4.2.2. Fotodiodo

Un fotodiodo es un semiconductor construido con una unión PN (Positivos o huecos, Negativos o electrones libres), sensible a la incidencia de la luz visible o infrarroja. Para que su funcionamiento sea correcto se polariza inversamente, con lo que se producirá una cierta circulación de corriente cuando sea excitado por la luz solar. Debido a su construcción, los fotodiodos se comportan como células fotovoltaicas, es decir, en ausencia de luz exterior generan una tensión muy pequeña con el positivo en el ánodo y el negativo

en el cátodo. Esta corriente que esta presente en ausencia de luz solar recibe el nombre de corriente de oscuridad.



b) Símbolo.

a) Físicamente.

FIGURA 5.5. FOTODIODO.

5.4.2.2.1. Operación del fotodiodo

Cuando una luz solar de suficiente energía llega al diodo, excita un electrón dándole movimiento y crea un hueco con carga positiva. Si la absorción ocurre en la zona de agotamiento de la unión, o a una distancia de difusión de él, estos portadores son retirados de la unión por el campo de la zona de agotamiento, produciendo una fotocorriente.

Los fotodiodos de avalancha tienen una estructura similar, pero trabajan con voltajes inversos mayores. Esto permite a los portadores de carga fotogenerados el ser multiplicados en la zona de avalancha del diodo, resultando en una ganancia interna, que incrementa la respuesta del dispositivo.

5.4.2.2.2. Composición del fotodiodo

El material empleado en la composición de un fotodiodo es un factor crítico para definir sus propiedades. Suelen estar compuestos de silicio (sensible a la luz visible) con una longitud de onda de hasta 1µm (micrómetro); germanio para luz infrarroja con una longitud de onda hasta aprox. 1,8 µm (micrómetro) o de cualquier otro material semiconductor.

Material	Longitud de onda nanómetros (nm)
Silicio	190–1100
Germanio	800–1700
Indio galio arsénico (InGaAs)	800–2600
sulfuro de plomo	<1000-3500

TABLA 5.1. MATERIALES DEL FOTODIODO.

También es posible la fabricación de fotodiodos para su uso en el campo de los infrarrojos medios (longitud de onda entre 5 y 20 µm), pero estos, requieren refrigeración con nitrógeno líquido.

5.4.2.3. Fotorresistencia

Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Puede también ser llamado fotorresistor, fotoconductor, célula fotoeléctrica o resistor dependiente de la luz cuyas siglas (LDR) se originan de su nombre en inglés light-dependent resistor.

Un **fotorresistor** está hecho de un semiconductor de alta resistencia. Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por la elasticidad del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre que resulta (y su hueco asociado) conduce electricidad, de tal modo que disminuye la resistencia.

FIGURA 5.6.

FOTORRESISTOR.

Un dispositivo fotoeléctrico puede ser intrínseco o extrínseco. En dispositivos intrínsecos, los únicos electrones disponibles están en la banda de la valencia, por lo tanto el fotón debe tener bastante energía para excitar el electrón a través de toda la banda prohibida. Los dispositivos extrínsecos tienen impurezas agregadas, que tienen energía de estado a tierra más cercano a la banda de conducción puesto que los electrones no tienen que saltar lejos, los fotones más bajos de energía (de mayor longitud de onda y frecuencia más baja) son suficientes para accionar el dispositivo.

5.4.2.3.1. Las células de sulfuro de cadmio

El sulfuro de cadmio o las células de sulfuro del cadmio (CdS) se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz solar que pulsa la célula. Cuanta más luz pulsa, más baja es la resistencia. Aunque no es exacta, incluso una célula simple de CdS puede tener una amplia gama de resistencia de cerca de 600 ohmios en luz brillante a 1 o 2 MΩ en oscuridad.

Las células son también capaces de reaccionar a una amplia gama de frecuencias, incluyendo infrarrojo (IR), luz visible, y ultravioleta (UV).

5.4.2.3.2. Usos de la fotorresistencia

Se fabrican de diversos tipos. Se pueden encontrar células baratas de sulfuro del cadmio en artículos de consumo, por ejemplo cámara fotográfica, medidores de luz, relojes con radio, alarmas de seguridad y sistemas de encendido y apagado del alumbrado de calles en función de la luz ambiente. En el otro extremo de la escala, los fotoconductores de Ge:Cu son los sensores que funcionan dentro de la gama más baja "radiación infrarroja".

5.4.2.4. Fototransistor

Se llama fototransistor a un transistor sensible a la luz, normalmente a los infrarrojos. La luz incide sobre la región de base, generando portadores en ella. Esta carga de base lleva el transistor al estado de conducción. El fototransistor es más sensible que el fotodiodo por el efecto de ganancia propio del transistor.

En el mercado se encuentran fototransistores tanto con conexión de base como sin ella y tanto en cápsulas plásticas como metálicas (TO-72, TO-5) provistas de una lente.

Se han utilizado en lectores de cinta y tarjetas perforadas, lápices ópticos, etc. Para comunicaciones con fibra óptica se prefiere usar detectores con fotodiodos p-i-n. También se pueden utilizar en la detección de objetos cercanos cuando forman parte de un sensor de proximidad.

Se utilizan ampliamente encapsulados conjuntamente con un LED, formando interruptores ópticos (opto-swich), que detectan la interrupción del haz de luz por un objeto. Existen en dos versiones: de transmisión y de reflexión.

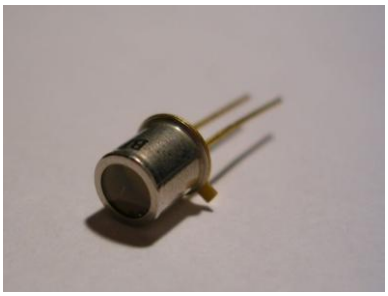


FIGURA 5.7. FOTOTRANSISTOR.

5.4.2.5. Optoacopladores

Un optoacoplador combina un dispositivo semiconductor formado por un fotoemisor (LED), un fotoreceptor (FOTODIODO) y entre ambos hay un camino por donde se transmite la luz, como se muestra en la figura 5.8. Todos estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP.

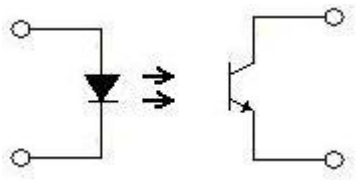


FIGURA 5.8. OPTOACOPLADOR.

5.4.2.5.1. Funcionamiento del Optoacoplador

La señal de entrada es aplicada al ***fotoemisor*** y la salida es tomada del fotoreceptor. Los optoacopladores son capaces de convertir una señal eléctrica en una señal luminosa modulada y volver a convertirla en una señal eléctrica. La gran ventaja de un optoacoplador reside en el aislamiento eléctrico que puede establecerse entre los circuitos de entrada y salida.

Los ***fotoemisores*** que se emplean en los optoacopladores de potencia son diodos que emiten rayos infrarrojos (IRED) y los fotoreceptores pueden ser tiristores o transistores. Cuando aparece una tensión sobre los terminales del diodo IRED, este emite un haz de rayos infrarrojo que transmite a través de una pequeña guía-ondas de plástico o cristal hacia el fotorreceptor. La energía luminosa que incide sobre el fotorreceptor hace que este genere

una tensión eléctrica a su salida. Este responde a las señales de entrada, que podrían ser pulsos de tensión.

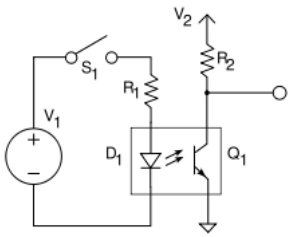


FIGURA 5.9. EL OPTOACOPLADOR COMBINA UN LED Y UN FOTODIODO.

5.4.3. DE FUERZA.

Se le llama fuerza a cualquier acción o influencia capaz de modificar el estado de movimiento o de reposo de un cuerpo, es decir, de imprimirle una aceleración modificando la velocidad, la dirección o el sentido de su movimiento.

5.4.3.1. Piezoelectricidad

La piezoelectricidad (del griego piezein, "estrujar o apretar") es un fenómeno presentado por determinados cristales que al ser sometidos a tensiones mecánicas adquieren una polarización eléctrica en su masa, apareciendo una diferencia de potencial y cargas eléctricas en su superficie. Este fenómeno también se presenta a la inversa, esto es, se deforman bajo la acción de fuerzas internas al ser sometidos a un campo eléctrico. El efecto piezoeléctrico es normalmente reversible: al dejar de someter los cristales a un voltaje exterior o campo eléctrico, recuperan su forma.

Los materiales piezoeléctricos son cristales naturales o sintéticos que no poseen centro de simetría. El efecto de una compresión o de un cizallamiento consiste en disociar los centros de gravedad de las cargas positivas y de las cargas negativas. Aparecen de este modo dipolos elementales en la masa y, por influencia, cargas de signo opuesto en las superficies enfrentadas.

5.4.3.1.1. Aplicaciones

Una de las aplicaciones más extendidas de este tipo de cristales son los encendedores electrónicos. En su interior llevan un cristal piezoeléctrico que es golpeado de forma brusca por el mecanismo de encendido. Este golpe seco provoca una elevada corriente eléctrica capaz de crear un arco voltaico o chispa que encenderá el mechero.

Otra de las importantes aplicaciones de un cristal piezoeléctrico es su utilización como sensor de vibración. Cada una de las variaciones de presión producidas por la vibración provoca un pulso de corriente proporcional a la fuerza ejercida, (esto ha convertido de una forma fácil una vibración mecánica en una señal eléctrica lista para amplificar). Basta con

conectar un cable eléctrico a cada una de las caras del cristal y enviar esta señal hacia un amplificador. Por ejemplo, en pastillas piezoeléctricas de guitarra.

5.4.4. DE DESPLAZAMIENTO

5.4.4.1. Potenciómetro

El potenciómetro original es un tipo de puente de circuito para medir voltajes. La palabra se deriva de "voltaje potencial" y "potencial" era usado para referirse a "fuerza". El potenciómetro original se divide en cuatro clases: el potenciómetro de resistencia constante, el potenciómetro de corriente constante, el potenciómetro microvolt y el potenciómetro termopar.

Se utiliza para medir voltajes debajo de 1.5 V. En este circuito, la tensión desconocida está conectado a través de una sección del alambre de la resistencia, los extremos de la cual están conectados con una célula electroquímica estándar que proporciona una corriente constante a través del alambre, fuerza electromotriz (fem) desconocida, en serie con un galvanómetro, entonces se conectada a través de una sección de longitud variable del alambre de la resistencia usando un contacto que se desliza. El contacto que se desliza se mueve hasta que ninguna corriente fluya dentro o fuera de la célula estándar, según lo indicado por un galvanómetro en serie con la fem desconocido, el voltaje a través de la sección seleccionada del alambre es entonces igual al voltaje desconocido.

Todo lo que queda es calcular el voltaje desconocido de la corriente y de la fracción de la longitud del alambre de la resistencia que fue conectado con la fem desconocido. El galvanómetro no necesita ser calibrado, pues su única función es leer cero. Cuando el galvanómetro lee cero, no se obtiene ninguna corriente de la fuerza electromotriz desconocida y así que la lectura es independiente de la resistencia interna de la fuente.

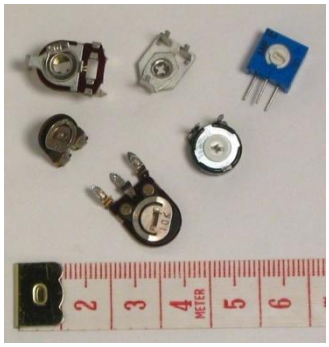


FIGURA 5.10. POTENCIÓMETRO.

5.4.4.2. Conmutadores

Los sensores digitales de desplazamiento más sencillos son los conmutadores. Éstos se usan de muchas maneras y es posible hacerlos funcionar en forma manual o conectarlos a mecanismos de alguna clase. Los conmutadores de funcionamiento manual incluyen los de palanca, que a menudo se usan como interruptores de conexión en equipos eléctricos, y los de botón de acción momentánea, como los que se usan en los teclados de los computadores. Quizá no sea evidente de inmediato que los conmutadores de este tipo son sensores de posición, pero es claro que producen un valor de salida que depende de la posición de la palanca o superficie de entrada, por lo que se trata de sensores binarios.

Cuando se conecta un conmutador a alguna forma de mecanismo, su acción como sensor de posición se hace más obvia. Una forma común de semejante dispositivo es de **microconmutador**, que permite que se le haga funcionar mediante alguna fuerza externa. Los microconmutadores se usan con frecuencia como **conmutadores de límite**, que avisan cuando un mecanismo ha llegado al final de su desplazamiento seguro. Los interruptores también se usan en gran cantidad de aplicaciones especializadas de medición de posición, como los sensores de nivel de líquido. La figura 5.11 muestra un sensor de este tipo. Aquí el conmutador se hace funcionar mediante un flotador que se eleva con el nivel de líquido hasta que llega aun nivel específico.

FIGURA 5.11. SENSOR DE POSICIÓN MEDIANTE CONMUTADORES.

5.4.4.3. Rejillas ópticas y otras técnicas de cuenta

Los codificadores de incremento y las rejillas ópticas emplean la cuenta de sucesos para determinar el desplazamiento.

Un sensor de aproximidad, es utilizar una rueda dentada ferromagnética cerca del sensor; con forme gira la rueda, los dientes pasan cerca del sensor, aumentando su inductancia.

Entonces, el sensor puede detectar el paso de cada diente y así determinar a distancia recorrida. Una gran ventaja de este sensor es su tolerancia a medios sucios.

Otro sensor para determinar desplazamiento es la utilización de rejillas ópticas, como lo muestra la figura que usa el optoconmutador ranurado que se analizó en la sección.

En este método se usa un disco que tiene cierta cantidad de oficios o ranuras espaciadas en forma equidistante alrededor de su perímetro. El disco y el optoconmutador se montan de manera que el borde del disco se encuentra dentro de la ranura del conmutador. Conforme gira el disco, los orificios o ranuras hacen que el optoconmutador esté periódicamente abierto o cerrado, produciendo un tren de pulsos con una frecuencia determinada por la velocidad de rotación. Se puede medir el ángulo de rotación contando el número de pulsos. Un método similar utiliza un sensor inductivo de proximidad en lugar del optoconmutador, así como un disco ferromagnético. Una vez más, el sensor detecta los orificios o ranuras del disco y los usa para medir la rotación angular.

5.4.4.3.1. Aplicación de sensores de desplazamiento con rejillas ópticas

Una aplicación básica de estos tipos de sensores es la capturación del movimiento de un **ratón mecánico** estándar, el cual se integra de:

1.- Al arrastrarlo sobre la superficie gira la bola.
2.- ésta a su vez mueve los rodillos ortogonales.
3.- éstos están unidos a unos discos de codificaciones ópticas, opacas pero perforadas.
4.- dependiendo de su posición pueden dejar pasar o interrumpir señales infrarrojas de un diodo LED.
5.- Estos pulsos ópticos son captados por sensores que obtienen así unas señales digitales de la velocidad vertical y horizontal actual para trasmitirse finalmente a la computadora.

FIGURA 5.12. FUNCIONAMIENTO DEL MOUSE INTERNO.

5.4.5. DE SONIDO

El sonido es desde el punto de vista físico, el movimiento ondulatorio en un medio elástico (normalmente el aire), debido a cambios rápidos de presión, generados por el movimiento vibratorio de un cuerpo sonoro. En general, se llama sonido a la sensación en el tímpano (órgano del oído) producida por este movimiento.

5.4.5.1. Micrófono

El micrófono es un transductor electroacústico. Su función es la de transformar (traducir) las vibraciones debidas a la presión acústica ejercida sobre su cápsula por las ondas sonoras en energía eléctrica.

FIGURA 5.13. MICRÓFONO.

5.5. ACTUADORES

Se denominan actuadores a aquellos elementos que pueden provocar un efecto sobre un proceso automatizado. Los actuadores son dispositivos capaces de generar una fuerza a partir de líquidos, de energía eléctrica y aire. El actuador recibe la orden de un regulador o controlador y da una salida necesaria para activar a un elemento final de control como lo son las válvulas.

Existen tres tipos de actuadores:

- Hidráulicos.
- Neumáticos.
- Eléctricos.

Los actuadores hidráulicos, neumáticos eléctricos son usados para manejar aparatos mecatrónicos. Por lo general, los actuadores hidráulicos se emplean cuando lo que se necesita es potencia, y los neumáticos son simples posicionamientos. Sin embargo, los hidráulicos requieren demasiado equipo para suministro de energía, así como de

mantenimiento periódico. Por otro lado, las aplicaciones de los modelos neumáticos también son limitadas desde el punto de vista de precisión y mantenimiento.

Los actuadores eléctricos también son muy utilizados en los aparatos mecatrónicos, como son los robots. Los servomotores de corriente alterna (ca) sin escobillas se utilizarán en el futuro como actuadores de posicionamiento preciso debido a la demanda de funcionamiento sin tantas horas de mantenimiento.

Los más usuales son:

- Cilindros neumáticos e hidráulicos.- Realizan movimientos lineales.
- Motores (actuadores de giro) neumáticos e hidráulicos.- Realizan movimientos de giro por medio de energía hidráulica o neumática.
- Válvulas.- Las hay de mando directo, motorizadas, electroneumáticas, etc. Se emplean para regular el caudal de gases y líquidos.
- Resistencias calefactoras.- Se emplean para calentar.
- Motores eléctricos.- Los más usados son de inducción, de continua, brushless y paso a paso.
- Bombas, compresores y ventiladores.- Movidos generalmente por motores eléctricos de inducción.

5.5.1. CALOR

El calor es una forma de energía asociada al movimiento de los átomos, moléculas y otras partículas que forman la materia. El calor puede ser generado por reacciones químicas (como en la combustión), nucleares (como en la fusión nuclear de los átomos de hidrógeno que tienen lugar en el interior del Sol), disipación electromagnética (como en los hornos de microondas) o por disipación mecánica (fricción). Su concepto está ligado al Principio Cero de la Termodinámica, según el cual dos cuerpos en contacto intercambian energía hasta que su temperatura se equilibre.

5.5.1.1. Calefactor o calentador resistivo

Aparato, normalmente eléctrico, que proporciona a una estancia o recipiente un flujo rápido de aire caliente continuo mediante un radiador que genera una fuente de calor y un ventilador que calienta rápidamente el aire y lo transmite a toda la estancia.

5.5.1.1.1. El efecto Joule.

Un calefactor eléctrico es un dispositivo que produce energía calorífica a partir de la eléctrica. El tipo más difundido es el calefactor eléctrico "resistivo", donde la generación del calor se debe al efecto Joule.

Otros calefactores y refrigeradores eléctricos menos conocidos son los "termoeléctricos", que intercambian calor mediante un principio más sofisticado: el efecto Peltier.

Sirve para obtener calor de una forma cómoda, rápida, limpia, compacta, económica y hasta elegante.

Entre las aplicaciones más conocidas del efecto Joule se tienen los elementos de las estufas para calentar el ambiente, los filamentos de los secadores para el pelo, las resistencias de las planchas para la ropa, las hornillas de las cocinas, las resistencias de tostadores y hornos industriales, los calentadores en los hervidores de agua y fermentadores, los alambres para evitar el congelamiento en refrigeradores y el empañamiento en vidrios de las ventanas traseras de automóviles, los calefactores en peceras e invernaderos, y muchísimas aplicaciones más.

5.5.2. LUZ

5.5.2.1. Diodos emisores de luz

Un LED, siglas en inglés de Light-Emitting Diode (diodo emisor de luz) es un dispositivo semiconductor (diodo) que emite luz cuasi-monocromática, es decir, con un espectro muy angosto, cuando se polariza de forma directa y es atravesado por una corriente eléctrica. El color (longitud de onda), depende del material semiconductor empleado en la construcción del diodo, pudiendo variar desde el ultravioleta, pasando por el espectro de luz visible, hasta el infrarrojo, recibiendo éstos últimos la denominación de IRED (Infra-Red Emitting Diode).

Las características de estos dispositivos son similares a las de otros diodos semiconductores, pero con diferentes voltajes de funcionamiento. La emisión de luz de un LED es aproximadamente proporcional a al corriente que pasa a través de él, un dispositivo pequeño típico podría tener un voltaje de funcionamiento de 2.0 V y una corriente máxima de 30 mA.

Los LED se pueden usar en forma individual o en dispositivos de elementos múltiples. Un ejemplo de este último es el **indicador visual de LED de siete segmentos**, el cual consiste en siete LED dispuestos. Cada elemento se puede ENCENDER o APAGAR en forma individual para exhibir una gama de patrones y poder representar los dígitos del 0 al 9.

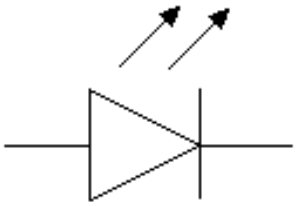
Los LED infrarrojos son ampliamente utilizados con fotodiodos o fototransistores para hacer posible la comunicación inalámbrica a corta distancia. Las variaciones en la corriente aplicada al LED se convierten en luz con una intensidad fluctuante que luego se convierte de nuevo en una señal eléctrica correspondiente mediante el dispositivo receptor. Esta técnica es muy utilizada en aplicaciones de control remoto para televisores y grabadoras de vídeo. En estos casos, la información transmitida se encuentra por lo general en forma digital. Como no hay conexión eléctrica entre el transmisor y el receptor, esta técnica se puede usar también para acoplar señales digitales entre dos circuitos que deben estar eléctricamente aislados. Esto se llama optoaislamiento. Existen pequeños optoaisladores autosuficientes que combinan la fuente de luz y el sensor en un solo envase. Las partes de entrada y salida de estos dispositivos están unidas sólo por la luz, lo cual permite que

produzca aislamiento entre los dos circuitos. Esto resulta particularmente útil cuando los dos circuitos están funcionando a niveles muy diferentes de voltaje. Los dispositivos típicos proporcionan aislamiento hasta 4 kV.

5.5.2.1.1. Funcionamiento

El funcionamiento físico consiste en que, un electrón pasa de la banda de conducción a la de valencia, perdiendo energía. Esta energía se manifiesta en forma de un fotón desprendido, con una amplitud, una dirección y una fase aleatoria.

b) Físicamente



a) Símbolo.

FIGURA 5.14. DIODO EMISOR DE LUZ LED.

5.5.2.1.2. Aplicaciones

Los diodos infrarrojos (IRED) se emplean desde mediados del siglo XX en mandos a distancia de televisores, habiéndose generalizado su uso en otros electrodomésticos como equipos de aire acondicionado, equipos de música, etc., y en general para aplicaciones de control remoto, así como en dispositivos detectores.

Los LED se emplean con profusión en todo tipo de indicadores de estado (encendido/apagado) en dispositivos de señalización (de tránsito, de emergencia) y en paneles informativos (el mayor del mundo, del NASDAQ, tiene 36.6 metros de altura y está en Times Square, Manhattan). También se emplean en el alumbrado de pantallas de cristal líquido de teléfonos móviles, calculadoras, agendas electrónicas, etc., así como en bicicletas y usos similares. Existen además impresoras LED.

FIGURA 5.15. PANTALLA DE LED`s EN EL ESTADIO DE LOS ARKANSAS RAZORBACKS.

El uso de lámparas LED en el ámbito de la iluminación (incluyendo la señalización de tráfico) es previsible que se incremente en el futuro, ya que aunque sus prestaciones son intermedias entre la lámpara incandescente y la lámpara fluorescente, presenta indudables ventajas: larga vida útil, una menor fragilidad y una mejor disipación de energía. Asimismo, para el mismo rendimiento luminoso, producen luz de color, mientras que los hasta ahora utilizados, tienen un filtro, lo que reduce notablemente su rendimiento.

FIGURA 5.16. LA PANTALLA EN FREEMONT STREET EN LAS VEGAS ES ACTUALMENTE LA MÁS GRANDE DEL MUNDO.

Los White LED`s son el desarrollo más reciente. Un intento muy bien fundamentado para sustituir las bombillas actuales por dispositivos mucho más eficientes desde un punto de vista energético.

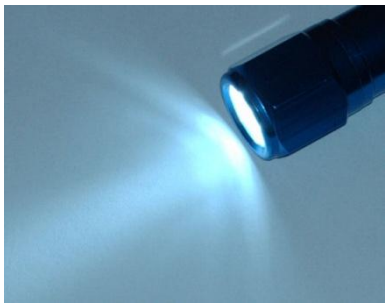


FIGURA 5.17. UNA PEQUEÑA LINTERNA A PILAS CON LED`s.

5.5.2.1.3. Conexión

La diferencia de potencial varía de acuerdo a las especificaciones relacionadas con el color y la potencia soportada.

En términos generales puede considerarse:

1. Rojo = 1 V
2. Rojo alta luminosidad = 1,9V
3. Amarillo = 1,7V a 3V
4. Verde = 2,4V
5. Verde alta luminosidad = 3,4V
6. Naranja = 2,4V
7. Blanco brillante = 3,4V
8. Azul = 3,4V
9. Azul 430nm = 4,6V
10. Blanco = 3,7V

Luego mediante la ley de Ohm, puede calcularse la resistencia adecuada para la tensión de la fuente que se utilizará.

$$R = \frac{V_{furnte} - (V_{d1} + V_{d2} + \cdots)}{I}$$

El término I (corriente), en la fórmula, se refiere al valor de corriente para la intensidad de luminosa que necesitamos. Lo común es de 10 V para LED`s de baja luminosidad y 20mA

para LED`s de alta luminosidad; un valor superior puede inhabilitar el LED o reducir de manera considerable su tiempo de vida.

Otros LED`s de una mayor capacidad de corriente conocidos como LED`s de potencia (1 w, 3 w, 5 w), pueden ser usados a 150 mA, 350 mA, 750 mA o incluso a 1,000 mA dependiendo de las características opto-eléctricas dadas por el fabricante.

Cabe recordar que también pueden conectarse varios en serie, sumándose las diferencias de potencial en cada uno.

También se pueden hacer configuraciones en paralelo, aunque este tipo de configuraciones no son muy recomendadas para diseños de circuitos con LED`s eficientes.

También se utilizan en la emisión de señales de luz que se trasmiten a través de fibra óptica.

5.5.2.2. Comunicación por fibra óptica

Para la comunicación a larga distancia no son adecuadas las sencillas técnicas utilizadas en las unidades de control remoto de los televisores, ya que les afecta en forma considerable la luz ambiental; es decir, la luz presente en el ambiente. Este problema se puede resolver mediante un cable de fibra óptica que capta la luz del transmisor y la pasa a lo largo del cable hasta el receptor, sin interferencia de las fuentes externas de luz. Por lo general, las fibras están hechas por un polímero óptico o de vidrio. Las primeras son económicas y fuertes, pero por su alta atenuación sólo son adecuadas para comunicaciones a corta distancia (alrededor de 20 metros).

Las fibras de vidrio tienen una atenuación mucho más baja y se puede usar a lo largo de cientos de kilómetros, pero son más caras que las fibras de polímetro, para las comunicaciones de larga distancia, la potencia disponible a partir de un LED infrarrojo convencional resulta insuficiente. En tales aplicaciones se pueden usar diodos laser. Éstos combinan las propiedades de emisión de luz de un LED con aplicación de luz de un láser para producir una fuente de luz coherente de alta potencia.

5.5.3. DESPLAZAMIENTO Y MOVIMIENTO

5.5.3.1. Motores

Los motores se pueden agrupar en tres amplias categorías: motores de corriente alterna (ca), motores de corriente directa (cd) y motores paso a paso, los **motores de ca** se usan sobre todo en aplicaciones de alta potencia y en aquellas en las que no se requiere gran precisión. El control de estos motores con frecuencia se efectúa mediante sencillas técnicas de apagado/encendido, aunque también se usan impulsores de potencia variable.

Los **motores de cd** son muy utilizados en sistemas de precisión de control de posición y otros sistemas electrónicos, en particular en aplicaciones de baja potencia. Estos motores tienen características muy definidas; su velocidad está determinada por el voltaje aplicado y su par se relaciona con la corriente. La gama de velocidades de los motores de cd puede ser muy amplia; algunos dispositivos pueden alcanzar velocidades que van desde decenas de miles de revoluciones por minuto hasta unas cuantas revoluciones por día. Algunos motores, en particular los de imán permanente, tienen un relación casi lineal entre velocidad y voltaje y entre el par y la corriente. Esto los hace muy fáciles de usar.

Los **motores paso a paso**, como su nombre lo indica, se mueven en pasos discretos. El motor consiste en un rotor central rodeado de varias bobinas o devanados.

5.5.3.2. Motor paso a paso (P-P)

El motor paso a paso se comporta de la misma manera que un convertidor digital-analógico y puede ser gobernado por impulsos procedentes de sistemas lógicos.

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

Estos motores poseen la habilidad de poder quedar fijos en una posición o bien totalmente libres. Si una o más de sus bobinas están energizadas, el motor estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.

En este trabajo trataremos solamente los motores paso a paso del tipo de imán permanente, ya que estos son los más usados en robótica.

FIGURA 5.18. MOTOR PASO A PASO.

5.5.3.3. Funcionamiento y estructura

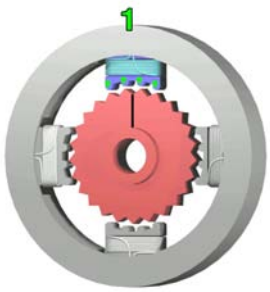
El motor paso a paso está constituido, como la mayoría de los motores eléctricos, esencialmente de dos partes:

Una parte fija llamada **"estator"**, construida a base de cavidades en las que van depositadas las bobinas, que excitadas convenientemente formarán los polos norte-sur de forma que se cree un campo magnético giratorio.

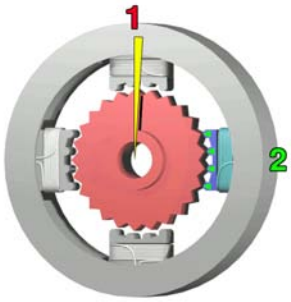
Una parte móvil, llamada **"rotor"** construida bien con un imán permanente o bien por un inducido ferromagnético, con el mismo número de pares de polos que el contenido en una sección de la bobina del estator; este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente.

Si por el medio del control que sea (electrónico, informático, etc.), conseguimos excitar el estator creando los polos norte sur (N-S), y se hace variar dicha excitación de modo que el campo magnético formado efectúe un movimiento giratorio, la respuesta del rotor será seguir el movimiento de dicho campo, produciéndose de este modo el giro del motor.

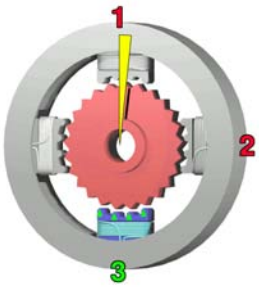
Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas están alimentadas, el motor estará fijo en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.



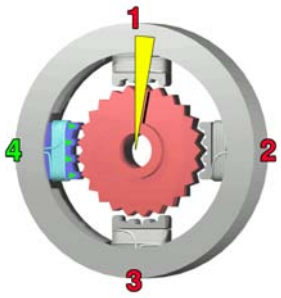
Paso 1.- la bobina 1 esta activada, atrayendo los cuatro dientes superiores imantados del rotor.



Paso 2.- la bobina 1 se apaga, y la bobina 2 (derecha) se activa, moviendo los dientes cercanos a la derecha. Resulta una rotación de 3.6°.



Paso 3.- De nuevo la bobina 2 se apaga, y la bobina 3 se activa. Resulta otra rotación de 3.6°.



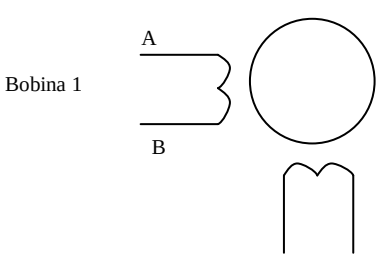
Paso 4.- La activación de la bobina 4 permite de nuevo la rotación de 3.6°. Cuando la bobina 1 se carge de nuevo, un diente habrá permutado su posición a la derecha; como hay 25 dientes, se necesitaran 100 pasos para un giro completo.

FIGURA 5.19. SECUENCIA DEL MOTOR PASO A PASO.

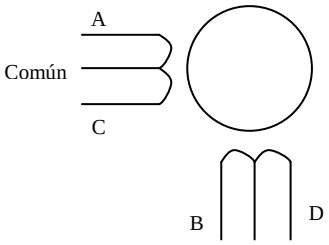
5.5.3.4. Tipos de motores paso a paso

Existen dos tipos de motores paso a paso de imán permanente:

- **Bipolar.-** Estos tiene generalmente cuatro cables de salida ver la figura 5.20 (a). Necesitan ciertos trucos para ser controlados, debido a que requieren del cambio de dirección del flujo de corriente a través de las bobinas en la secuencia apropiada para realizar un movimiento.
- **Unipolar.-** Estos motores suelen tener 6 o 5 cables de salida, dependiendo de su conexionado interno ver la figura 5.20. b). Este tipo se caracteriza por ser más simple de controlar.



a) Motor P-P Bipolar



b) Motor P-P Unipolar

FIGURA 5.20. MOTOR PASO A PASO.

5.5.3.5. Secuencias para manejar motores paso a paso Unipolares

Existen tres secuencias posibles para este tipo de motores, las cuales se detallan a continuación. Todas las secuencias comienzan nuevamente por el paso 1 una vez alcanzado el paso final (4 u 8). Para revertir el sentido de giro, simplemente se deben ejecutar las secuencias en modo inverso.

5.5.3.6. Secuencia Normal

Esta es la secuencia como se muestra en la tabla 5.2 es la más usada y la que generalmente recomienda el fabricante. Con esta secuencia el motor avanza un paso por vez y debido a que siempre hay al menos dos bobinas activadas, se obtiene un alto torque de paso y de retención.

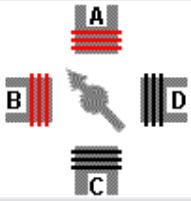
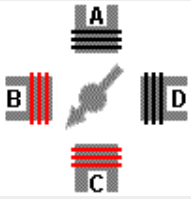
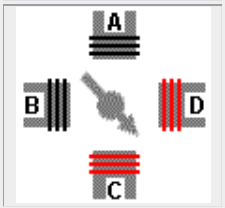
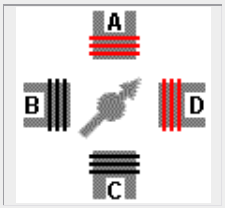
PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	**ON**	**ON**	OFF	OFF	
2	OFF	**ON**	**ON**	OFF	
3	OFF	OFF	**ON**	**ON**	
4	**ON**	OFF	OFF	**ON**	

TABLA 5.2. SECUENCIA NORMAL.

5.5.3.7. Secuencia del tipo wave drive.

En esta secuencia se activa solo una bobina a la vez. En algunos motores esto brinda un funcionamiento más suave. La contrapartida es que al estar solo una bobina activada, el torque de paso y retención es menor.

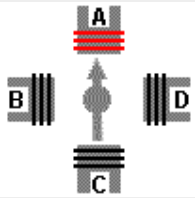
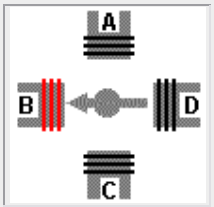
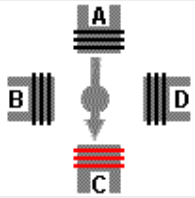
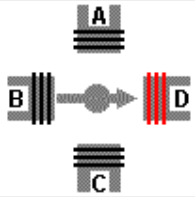
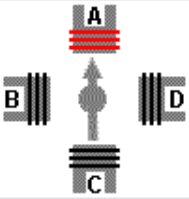
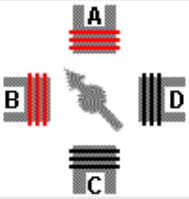
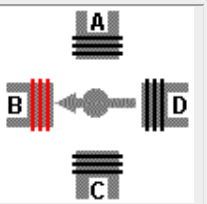
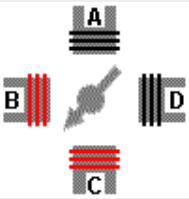
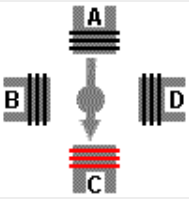
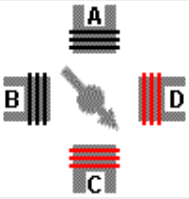
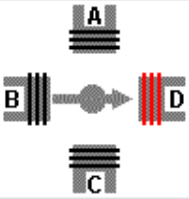
PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	**ON**	OFF	OFF	OFF	
2	OFF	**ON**	OFF	OFF	
3	OFF	OFF	**ON**	OFF	
4	OFF	OFF	OFF	**ON**	

TABLA 5.3. SECUENCIA WAVE DRIVE.

5.5.3.8. Secuencia del tipo medio paso

En esta secuencia se activan las bobinas de tal forma de brindar un movimiento igual a la mitad del paso real. Para ello se activan primero 2 bobinas y luego solo 1 y así sucesivamente. Como vemos en la tabla siguiente, la secuencia completa consta de 8 movimientos en lugar de 4.

PASO	Bobina A	Bobina B	Bobina C	Bobina D	
1	**ON**	OFF	OFF	OFF	
2	**ON**	**ON**	OFF	OFF	
3	OFF	**ON**	OFF	OFF	
4	OFF	**ON**	**ON**	OFF	
5	OFF	OFF	**ON**	OFF	
6	OFF	OFF	**ON**	**ON**	
7	OFF	OFF	OFF	**ON**	

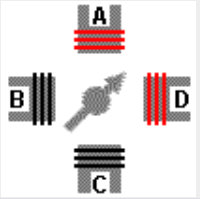
8	**ON**	OFF	OFF	**ON**	

TABLA 5.4. SECUENCIA TIPO MEDIO PASO.

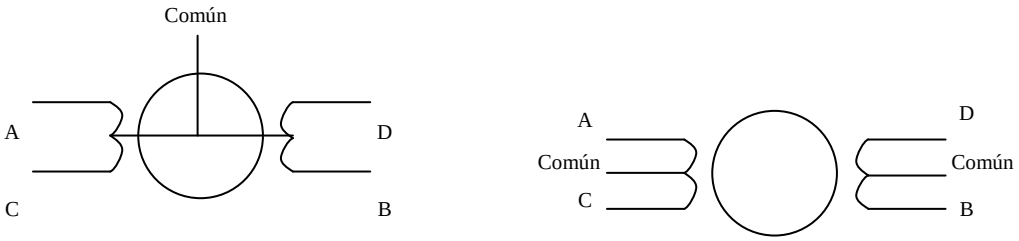
Cabe destacar que debido a que los motores paso a paso son dispositivos mecánicos y como tal deben vencer ciertas inercias, el tiempo de duración y la frecuencia de los pulsos aplicados es un punto muy importante a tener en cuenta. En tal sentido el motor debe alcanzar el paso antes que la próxima secuencia de pulsos comience. Si la frecuencia de pulsos es muy elevada, el motor puede reaccionar en alguna de las siguientes formas:

- Puede que no realice ningún movimiento en absoluto.
- Puede comenzar a vibrar pero sin llegar a girar.
- Puede girar erráticamente.
- puede llegar a girar en sentido opuesto.

Para obtener un arranque suave y preciso, es recomendable comenzar con una frecuencia de pulso baja y gradualmente ir aumentándola hasta la velocidad deseada sin superar la máxima tolerada. El giro en reversa debería también ser realizado previamente bajando la velocidad de giro y luego cambiar el sentido de rotación.

5.5.3.9. Identificación de los cables en Motores Paso a Paso (Unipolares)

Cuando se trabaja con motores paso a paso usados o bien nuevos, pero de los cuales no existen hojas de datos. Es posible averiguar la distribución de los cables a los bobinados y el cable común en un motor de paso unipolar de 5 o 6 cables.



b) Motor P-P con 5 cables de salida

a) Motor P-P con 6 cables de salida

FIGURA 5.21. IDENTIFICACIÓN DE CABLES.

Para identificación de cables de un motor paso a paso se siguen las siguientes instrucciones:

1.- Aislando el cable(s) común que va a la fuente de alimentación: como se aprecia en las figuras anteriores, en el caso de motores con 6 cables, estos poseen dos cables comunes, pero generalmente poseen el mismo color, por lo que lo mejor es unirlos antes de comenzar las pruebas.

Usando un multimetro para verificar la resistencia entre pares de cables, el cable común será el único que tenga la mitad del valor de la resistencia entre ella y el resto de los cables.

Esto es debido a que el cable común tiene una bobina entre ella y cualquier otro cable, mientras que cada uno de los otros cables tienen dos bobinas entre ellos. De ahí la mitad de la resistencia medida en el cable común.

2.- Identificando los cables de las bobinas (A, B, C y D): aplicar un voltaje al cable común (generalmente 12 volts, pero puede ser más o menos) y manteniendo uno de los otros cables a masa (GND) mientras vamos poniendo a masa cada uno de los demás cables de forma alternada y observando los resultados. El proceso se puede apreciar en la siguiente tabla:

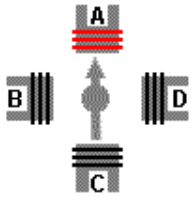
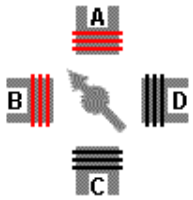
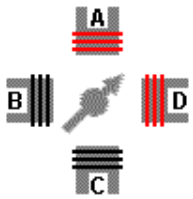
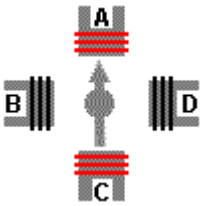
Seleccionar un cable y conectarlo a masa. Ese será llamado cable **A**.	
Manteniendo el cable **A** conectado a masa, probar cuál de los tres cables restantes provoca un paso en sentido antihorario al ser conectado también a masa. Ese será el cable **B**.	
Manteniendo el cable **A** conectado a masa, probar cuál de los dos cables restantes provoca un paso en sentido horario al ser conectado a masa. Ese será el cable **D**.	
El último cable debería ser el cable **C**. Para comprobarlo, basta con conectarlo a masa, lo que no debería generar movimiento alguno debido a que es la bobina opuesta a la **A**.	

TABLA 5.5. SECUENCIA TIPO MEDIO PASO.

Nota:

La nomenclatura de los cables (A, B, C, D) es totalmente arbitraria.

Un motor de paso con 5 cables corresponde al tipo unipolar ya que tiene 4 fases y un común.

Un motor de paso con 6 cables también corresponde a unipolar ya que contiene 4 fases y 2 cables comunes para alimentación. Pueden ser del mismo color.

Un motor de pasos con solo 4 cables es comúnmente bipolar.

5.6. CONTROL DE UN MOTOR PASO A PASO CON UNA PC

Normalmente una PC siempre contiene un puerto paralelo y probablemente está conectada una impresora o un Scanner. Si se desconecta el cable se verá que tiene 25 contactos (pines), y su nombre es "Conector DB25".

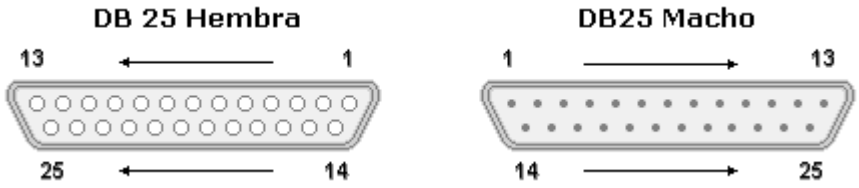


FIGURA 5.22. CONECTOR DB25.

Cada pin de los conectores tiene un número asignado por lo cual es muy importante que al armar los propios cables se identifiques correctamente, pare evitar el mal funcionamiento.

Este puerto dispone de tres registros de 8 bit cada uno es decir un byte.

Cada uno de estos registros, se denominan puertos o PORT., y cada uno de sus bits, representa un pin determinado del puerto. Los pin`s que van del 18 al 25, incluyendo a ellos dos, son para masa, y sirven para conectar las descargas de los circuitos.

Como se menciono el puerto paralelo contiene tres registros que son los siguientes:

1.- **Puerto de datos (Pin 2 al 9)**: Es el PORT 888 y dirección en el BIOS (0x378h), es de solo escritura, por este registro enviaremos los datos al exterior de la PC, cuidado...!!!, no envíes señales eléctricas al ordenador por estos pines.

2.- **Puerto de estado (Pin 15, 13, 12, 10 y 11)**: Es el PORT 889 y dirección en el BIOS (0x379h), es de solo lectura, por aquí enviaremos señales eléctricas al ordenador, de este registro solo se utilizan los cinco bits de más peso, que son el bit 7, 6, 5, 4 y 3 teniendo en cuenta que el bit 7 funciona en modo invertido.

3.- **Puerto de control (Pin 1, 14, 16 y 17)**: Es el correspondiente al PORT 890, y dirección en el BIOS (0x37Ah), es de lectura/escritura, es decir, podremos enviar o recibir señales eléctricas, según nuestras necesidades. De los 8 bits de este registro solo se utilizan los cuatro de menor peso o sea el 0, 1, 2 y 3, con un pequeño detalle, los bits 0, 1, y 3 están invertidos.

En la siguiente imagen puedes ver los tres registros, sus bits y los pines asignados a cada uno de ellos. La figura corresponde a un conector DB-25 (Hembra).

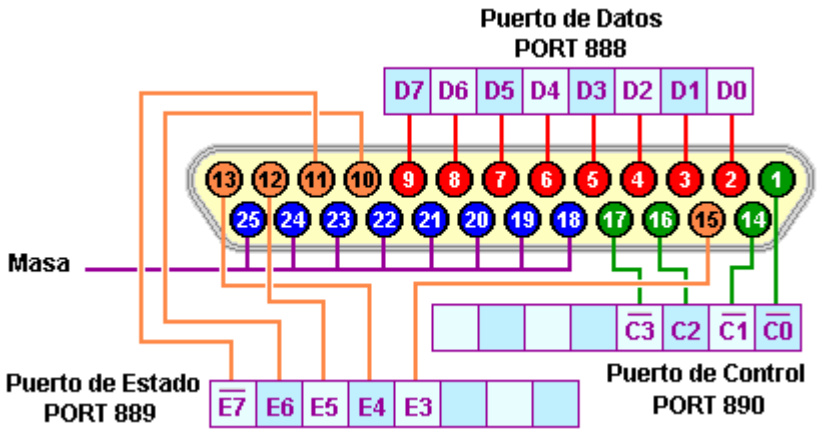


FIGURA 5.23. REGISTROS DEL PUERTO PARALELO.

Es importante conocer la tensión de trabajo del puerto, ya que corresponde a 5 voltios, por lo que requiere una fuente estabilizada o regulada de tensión, esto es importante tenerlo en cuenta, ya que estaremos enviando señales al puerto. Por otro lado, si bien se puede utilizar la PC para enviar señales al exterior sin necesitad de una fuente externa, es recomendable utilizarla y de esa manera no se exige demasiado al puerto y se evitan riesgos problemas.

Además si se activa un bit (por medio de un lenguaje de programación) de salida por el puerto, este permanecerá así hasta que se cambie, es decir que estará enviando 5V de forma continua hasta que se ponga a 0 (desactivar el bit).

A continuación se pasa al armado del circuito utilizando diferentes componentes y diseños para el control y funcionamiento del puerto paralelo.

5.6.1. ENVIÓ DE DATOS POR EL PUERTO PARALELO

Para iniciar primero se construirán unos circuitos que nos permita enviar señales por el puerto paralelo utilizando diodos LED`s, para la verificación de las señales de salida, conectados directamente al puerto, sin embargo es recomendable utilizar algunos componentes o buffers para asegurar el funcionamiento correcto como el consumo de los LED`s principalmente por el consumo de los LED`s que es superior al que nos envía el puerto paralelo.

5.6.2. UTILIZANDO EL BUFFER 74HCT245

Al utilizar un buffer como el 74HC244 o el 74HCT245, es recomendable este último ya que la construcción de la placa es más práctica, los datos del circuito integrado son los siguientes.

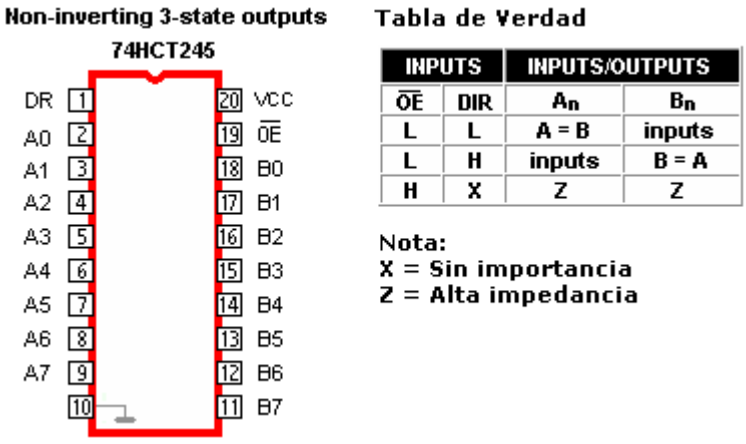


FIGURA 5.24. BUFFER 74HCT245.

Este integrado tiene la ventaja de ser bidireccional, es decir todos los pin`s A pueden ser entradas y los B salidas. En la siguiente tabla tienes los nombres de los pines y sus funciones correspondientes.

No. PIN	Nombre	Función
1	DIR	Control de dirección
2, 3, 4, 5, 6, 7, 8, 9	A0-A7	Entrada/Salida de datos
10	GND	Fuente (0V)
18, 17, 16, 15, 14, 13, 12, 11	B0-B7	Entrada/Salida de datos
19	$\overline{OE}$	Habilitación (Activo (L))
20	Vcc	Fuente (+5V)

TABLA 5.6. SECUENCIA TIPO MEDIO PASO.

En el siguiente esquema (figura 5.25) no se representaron todos los pines del puerto, sino los correspondientes al puerto de datos y los de masa, que son los que se utiliza en este experimento.

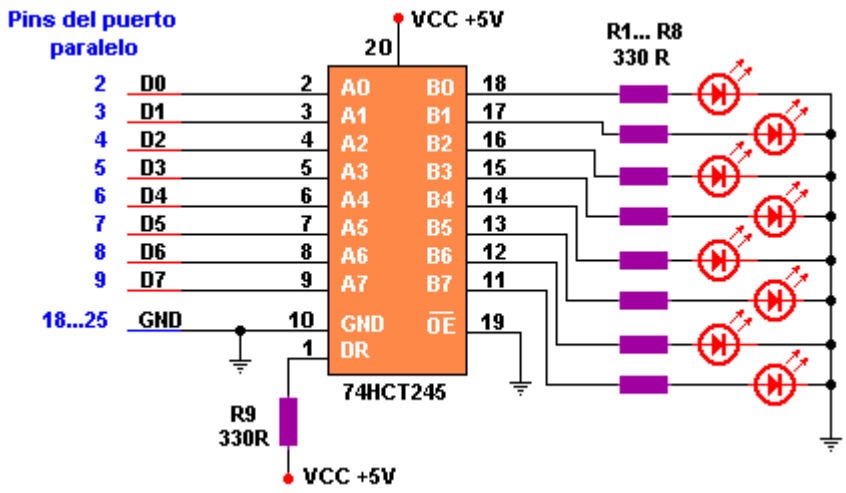


FIGURA 5.25. BUFFER 74HCT245 CON ENTRADAS Y SALIDAS.

Para su funcionamiento del circuito anterior se utiliza las siguientes sentencias o instrucciones en el lenguaje de programación C, donde realiza un barrido de encendido y apagado de los LED`s mostrados en la figura anterior, el código fuente en lenguaje C se muestra es el figura 5.26.

Cabe mencionar que la señales que mandaría a un motor paso a paso toma únicamente cuatro salidas del puerto paralelo de la PC, ya que el motor tiene cuatro fases como se mencionó anteriormente.

```
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#define LPT 0x378

typedef short _stdcall (*inpfuncPtr)(short portaddr);
    typedef void _stdcall (*oupfuncPtr)(short portaddr, short datum);

int main(void)
{
    HINSTANCE hLib=LoadLibrary("inpout32.dll");
    inpfuncPtr inp32=(inpfuncPtr) GetProcAddress(hLib, "Inp32");
    oupfuncPtr oup32=(oupfuncPtr) GetProcAddress(hLib, "Out32");


    //x = (inp32)(LPT);
    //(oup32)(LPT,x);

     int a[4]={1,2,4,8};
     int rep,i,res;

     printf("Ingresa cuantas veces se repetirá el encendido de los LED`s? \n");
     scanf("%d",&rep);

     for (i=0;i<rep;i++)
     {
     res=i%4;
     printf("%d\n",a[res]);
     (oup32)(LPT, a[res]);
     Sleep(250);

     }
     FreeLibrary(hLib);
      system("PAUSE");
     return 0;
}
```

FIGURA 5.26. CÓDIGO FUENTE PARA LAS SEÑALES DE SALIDA DEL PUERTO PARALELO.

Con el programa anterior, y conectando el circuito en el puerto paralelo de la PC se puede manipular los LED´s que se muestra e la siguiente figura, que contiene 4 LED´s , 4 resistencias de 330 Ω en la parte de atrás de los LED´s insertados en la protoboard y conectados a la tierra del puerto paralelo

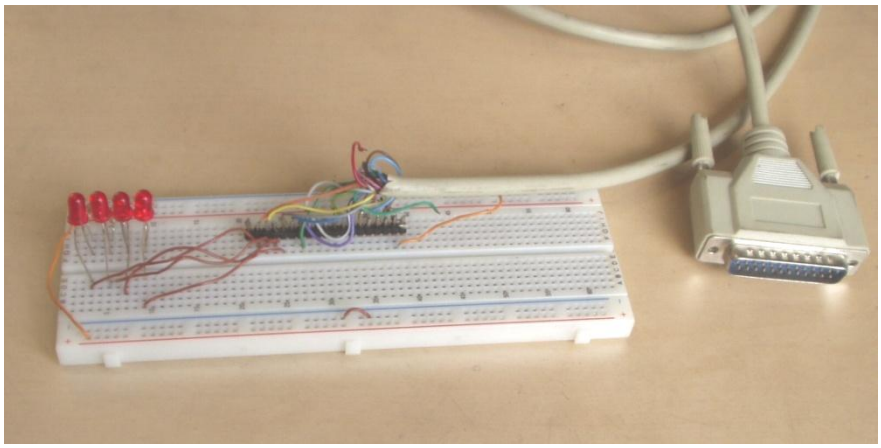


FIGURA 5.27. CIRCUITO SENCILLO UTILIZANDO LED´s Y RESISTENCIAS.

Si se desea encender algún aparato en la industria, en el hogar o en algún otro lugar, lo único que hay que hacer es agregar una etapa de potencia a cada salida, como se muestra en el siguiente diagrama, figura 5.28.

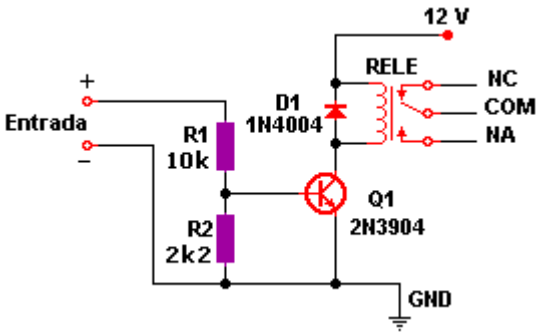


FIGURA 5.28. CIRCUITO PARA MANIPULAR UN APARATO EXTERNO.

5.6.3. RECIBIR DATOS EN EL PUERTO PARALELO

Para recibir datos en el computador por el puerto paralelo se utiliza un registro de estado, y solo dispones de 5 bits como se vio en la figura y sus pin`s son: 7, 6, 5, 4 y 3.

Hay que tener en cuenta que solo un bit puede tener dos estados posibles, ya sea 0 (0 voltios) o 1 (5 voltios) no podemos dejarlo librado al azar, razón por la cual, si no se envía señal alguna, éste deberá estar unido a masa.

El diagrama eléctrico es el siguiente;

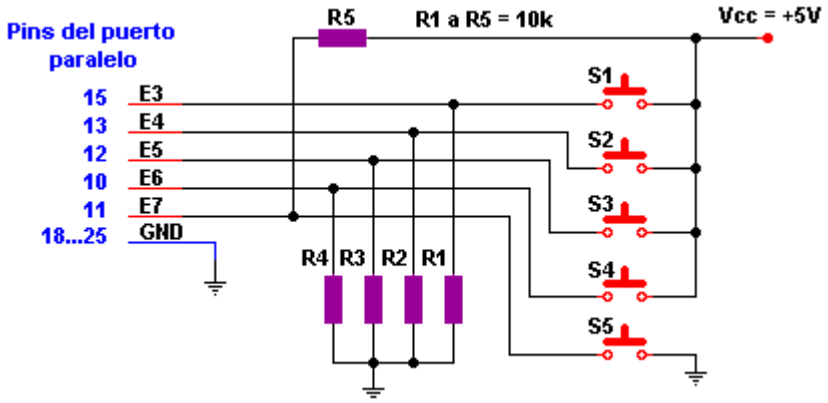


FIGURA 5.29. CIRCUITO PARA RECIBIR DATOS EN EL PUERTO PARALELO.

En el bit 7 la conexión es invertida si lo pones a 0 el programa lo leerá como un 1, si no presionas el pulsador se leerá como 0 lógico.

5.6.4. UTILIZANDO UN DRIVER DE CUATRO CANALES (L293B)

El L293B es un driver de 4 canales capaz de proporcionar una corriente de salida de hasta 1A por canal. Cada canal es controlado por señales de entrada compatibles TTL y cada pareja de canales dispone de una señal de habilitación que desconecta las salidas de los mismos. Dispone de una patilla para la alimentación de las cargas que se están controlando, de forma que dicha alimentación es independiente de la lógica de control.

La Figura que a continuación se indica el encapsulado de 16 pines, la distribución de patillas y la descripción de las mismas.

Pin Nombre Descripción Patillaje
1 Chip Enable 1 Habilitación de los canales 1 y 2
2 Input 1 Entrada del Canal 1
3 Output 1 Salida del Canal 1
4 GND Tierra de Alimentación
5 GND Tierra de Alimentación
6 Output 2 Salida del Canal 2
7 Input 2 Entrada del Canal 1
8 Vs Alimentación de las cargas
9 Chip Enable 2 Habilitación de los canales 3 y 4
10 Input 3 Entrada del Canal 3
11 Output 3 Salida del Canal 3
12 GND Tierra de Alimentación
13 GND Tierra de Alimentación
14 Output 4 Salida del Canal 4
15 Input 4 Entrada del Canal 4

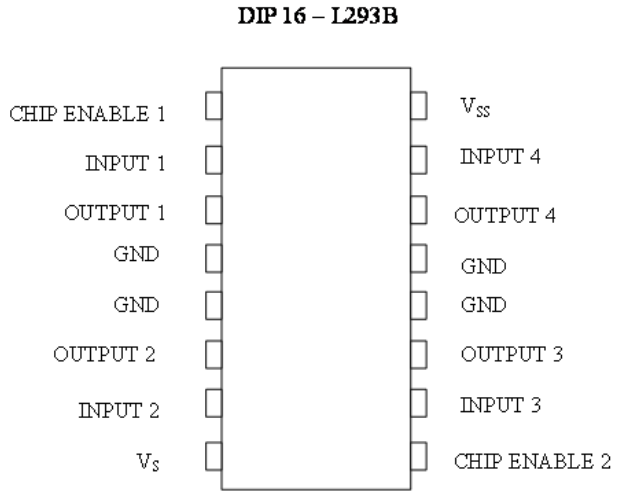
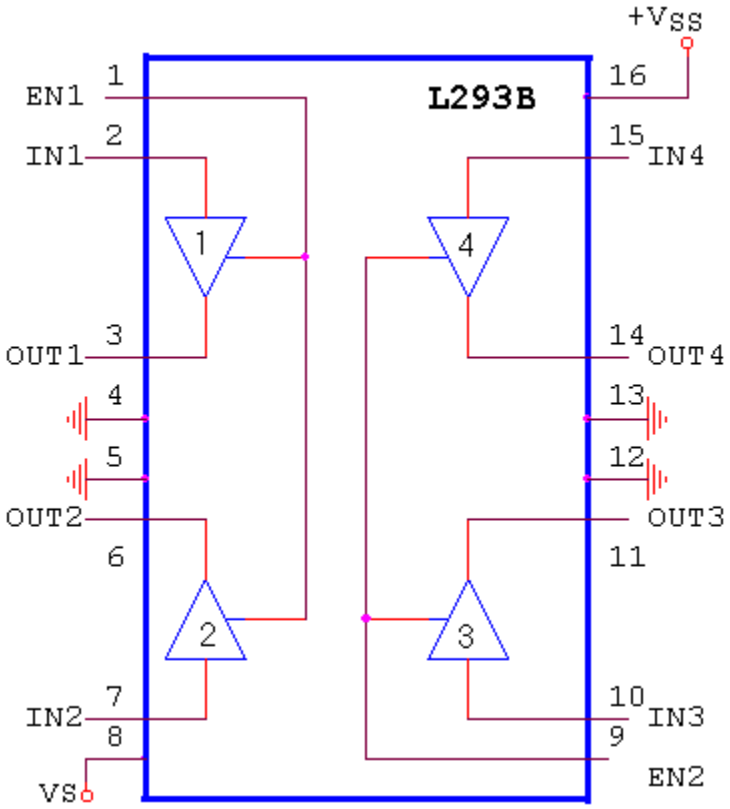


FIGURA 5.30. L293B.

5.6.4.1. Diagrama de bloques

En la Figura 5.31, se muestra el diagrama de bloques del L293B. La señal de control EN1 activa la pareja de canales formada por los drivers 1 y 2. La señal EN2 activa la pareja de drivers 3 y 4. Las salidas OUTN se asocian con las correspondientes OUTn. Las señales de salida son amplificadas respecto a las de entrada tanto en tensión (hasta+Vss) como en corriente (máx. 1 A).



VINn	VOUTn	VENn
H	H	H
L	L	H
H	Z	L
L	Z	L

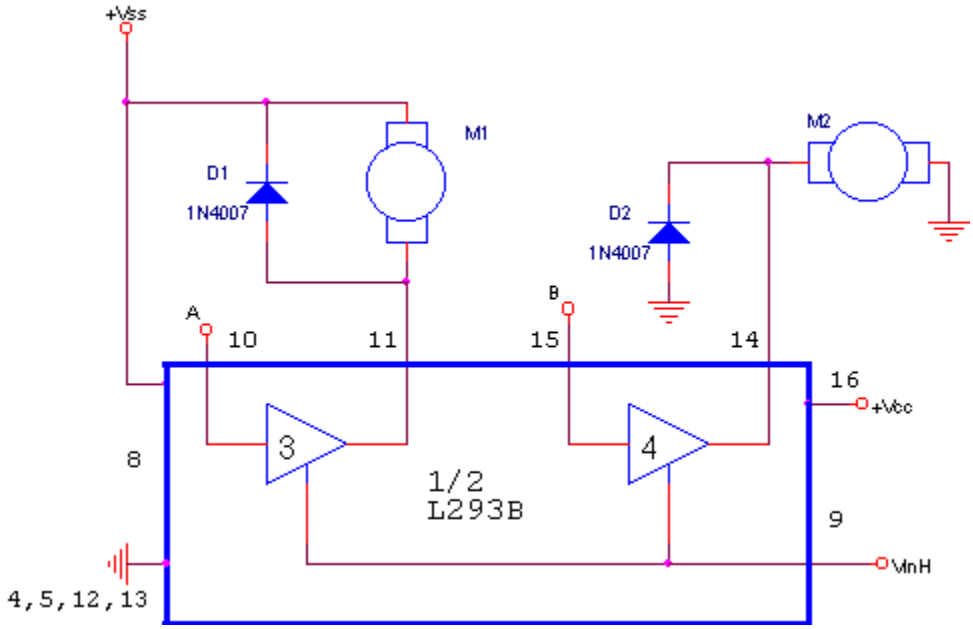
Donde:
H=Nivel alto "
L=Nivel bajo "
Z=Alta Impedan

FIGURA 5.31. DIAGRAMA DE BLOQUES DEL L293B.

5.6.4.2. Aplicación para girar dos motores en un solo sentido.

En la figura 5.32, se muestra el modo de funcionamiento de dos motores de corriente continua que giran en un único sentido.

- El motor M1 se activa al poner a nivel bajo la entrada de control A.
- El motor M2 se activa al poner a nivel alto la entrada de control B.



Vinh	A	M1	B	M2
H	H	Parada rápida del motor	H	Giro
H	L	Giro	L	Parada rápida del motor
L	X	Motor desconectado, giro libre	X	Motor desconectado, giro libre

FIGURA 5.32. CONEXIÓN DE 2 MOTORES DE CONTINUA M1 ACTIVADO POR "0" Y M2 POR "1".

Los diodos D1 y D2, están conectados para proteger el circuito cuando se generan los picos de arranque de los motores. Si no se trabaja a máxima potencia de trabajo pueden eliminarse del circuito.

5.6.4.3. Control del giro de un motor en los dos sentidos

El circuito de la figura 5.33, permite controlar el doble sentido de giro del motor. Cuando la entrada C está a nivel bajo y la D a nivel alto, el motor gira hacia la izquierda. Cambiando la entrada C a nivel alto y la D a nivel Bajo, se cambia el sentido de giro del motor hacia la derecha.

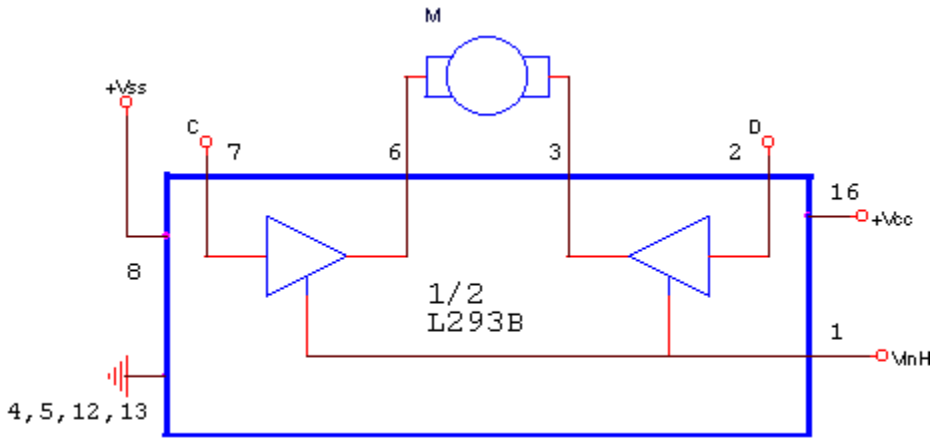


FIGURA 5.33. CIRCUITO DE CONTROL PARA EL DOBLE GIRO DE UN MOTOR DE CORRIENTE CONTINUA.

Si se quiere proteger el circuito contra posibles picos de corriente inversa cuando se arranca el motor, se recomienda conectar unos diodos tal y como se muestra en la figura

En este caso la tabla de funcionamiento es la siguiente:

Vinh	A	B	M
H	L	L	Parada rápida del motor
H	H	H	Parada rápida del motor
H	L	H	Giro a la Izquierda
H	H	L	Giro a la derecha
L	X	X	Motor desconectado

TABLA 5.7. DIAGRAMA DE BLOQUES DEL L293B.

5.6.4.4. Control de un motor paso a paso bipolar

En la Figura 5.34, se muestra una forma de conectar un motor bipolar paso a paso utilizando el circuito integrado L293. En este caso habrá que generar la secuencia adecuada al motor paso a paso para poder excitar de forma correcta las bobinas. La forma de proteger el circuito contra las corrientes que se producen en el momento de arranque del motor sería el mismo que el de la figura, pero utilizando para cada una de las bobinas del motor paso a paso un diodo.

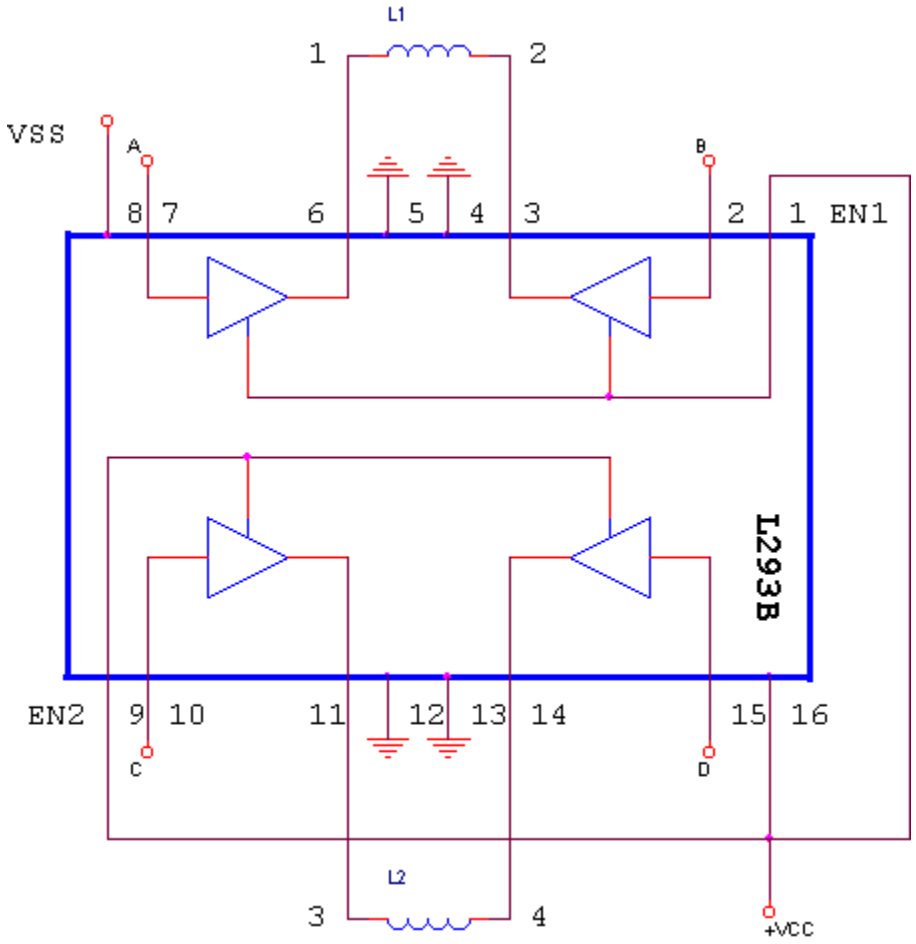


FIGURA 5.34. CONEXIÓN DE UN MOTOR PASO A PASO BIPOLAR AL L293B.

5.6.5. LA INTERFAZ DEL PROGRAMA PARA EL MANEJO DEL MOTOR PASO A PASO

Para llevar a cabo este control de los motores los parámetros del control de movimiento se da en al siguiente programa realizado en lenguaje C, figura 5.35. Encargado de realizar el control del motor paso a paso, la medida para el giro de un motor se puede dar en grados, sin embargo para este caso como en algunos otros, es preferible utilizar la medida en radianes

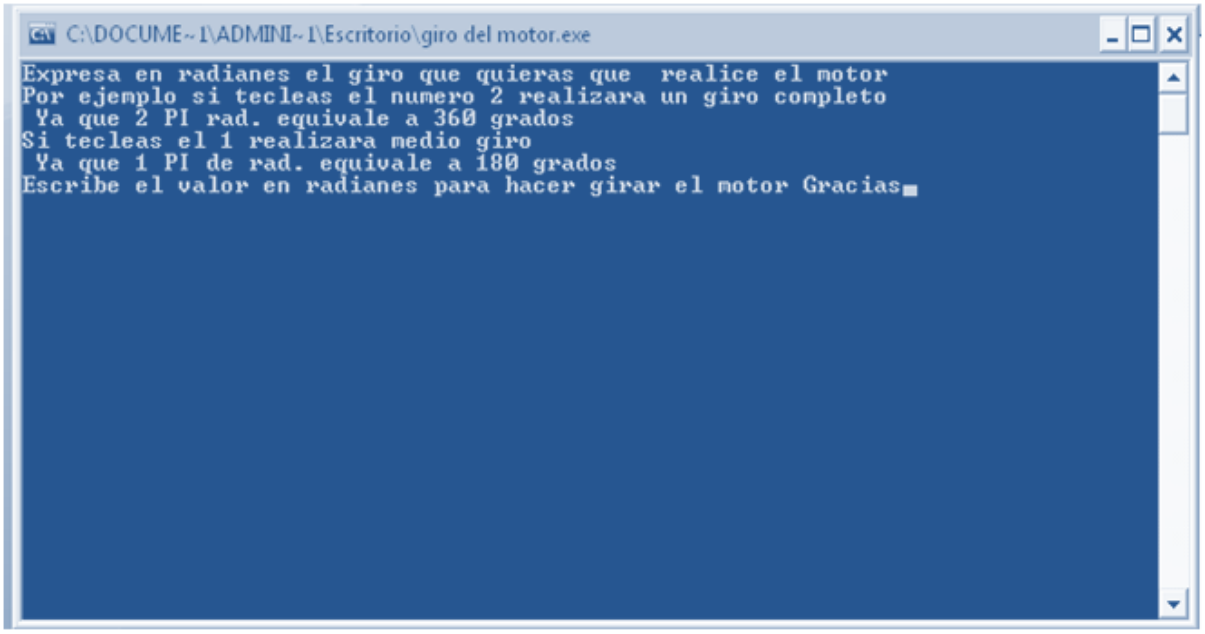
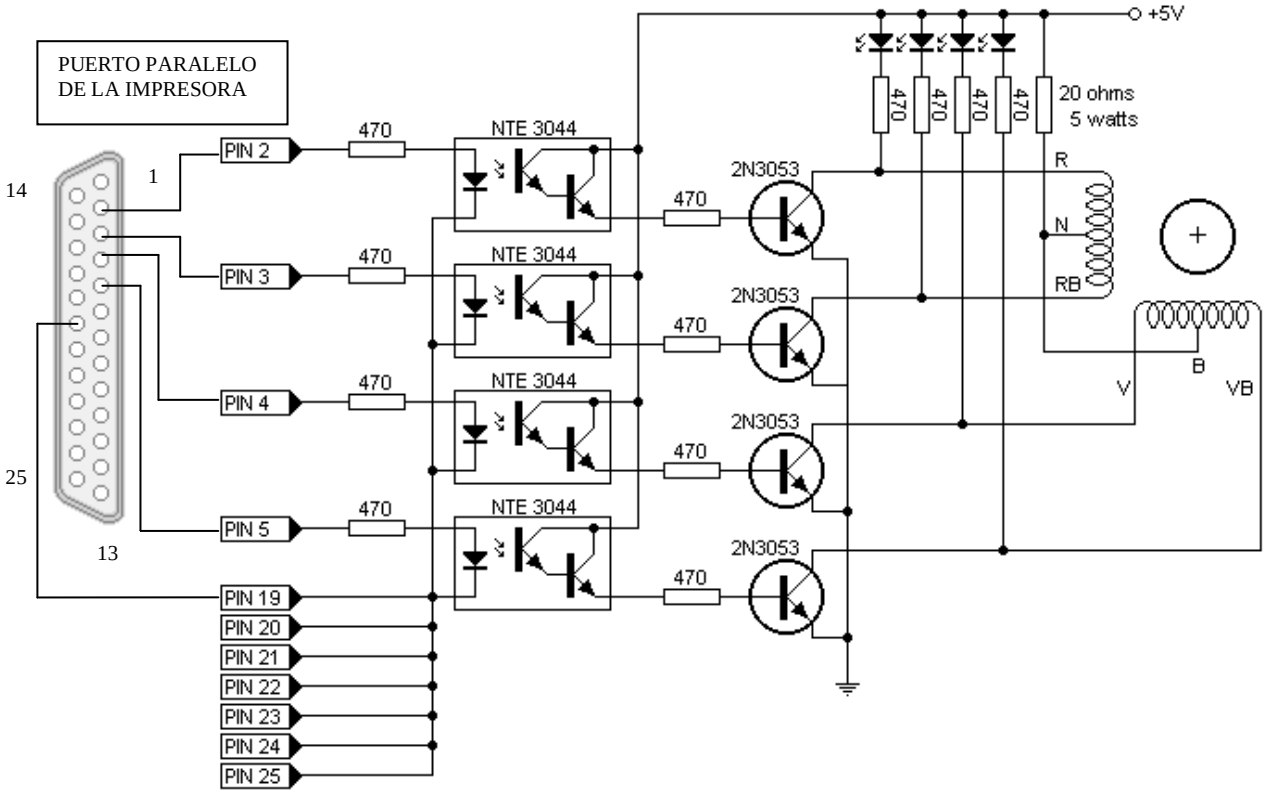


FIGURA 5.35. INTERFAZ DEL CONTROL DEL MOTOR PASO A PASO.

5.6.6. UTILIZANDO OPTOACOPLADORES Y BUFFERS PARA EL MANEJO DEL MOTOR PASO A PASO

Cuando se necesita precisión a la hora de mover un eje nada mejor que un motor paso a paso. Estos motores, a diferencia de los motores convencionales, no giran cuando se les aplica corriente si no se hace en la secuencia adecuada. El presente circuito de la figura 5.36, permite adaptar los niveles de potencia presentes en el puerto paralelo de una PC para poder manejar cómodamente un motor paso a paso por medio de un programa que ha sido desarrollado en el lenguaje de programación C.



5.36. CONTROL DE UN MOTOR PASO A PASO CON OPTOACOPLADORES.

La primera etapa del circuito se encarga de aislar la entrada proveniente de la PC por medio de optoacopladores. La segunda etapa consiste en buffer de corriente, que permite manejar las bobinas del motor. Las resistencias de 470 ohms junto con los diodos LED permiten monitoriar el adecuado funcionamiento del sistema. Este circuito se representa con los colores de los cables:

R = Cable Rojo
N = Cable Negro
RB = Cable Rojo y Blanco
V = Cable Verde
B = Cable Blanco
VB = Cable Verde y Blanco

5.6.7. UTILIZANDO UN DRIVER DE OCHO CANALES (ULN2803)

El siguiente diagrama (figura 5.37.), podemos apreciar otro ejemplo de conexión para controlar un motor paso a paso unipolar mediante el uso de un ULN2803, el cual es un arreglo (array) de 8 transistores tipo Darlington diseñados para manejar cargas de hasta 500 mA. Las entradas de activación (Activa A, B, C y D) son directamente activadas por la computadora.

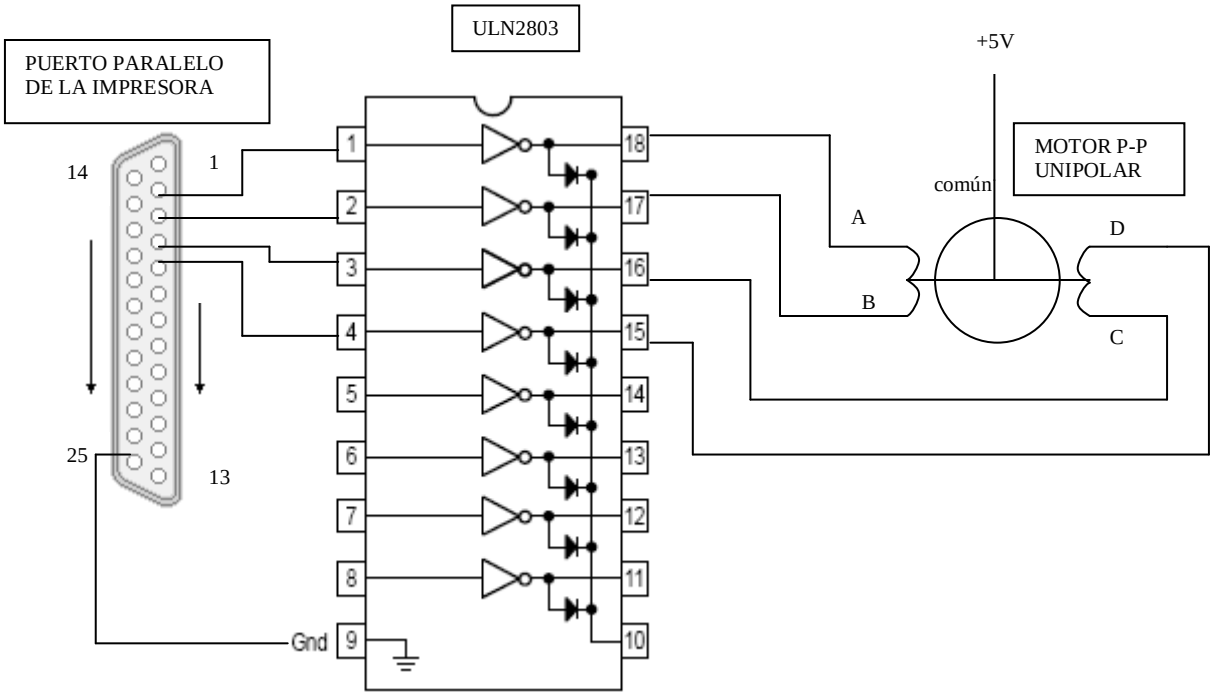


FIGURA 5.37. ULN2803 CONECTADO AL PUERTO DE PARALELO DE LA PC Y AL MOTOR PASO A PASO.

Cada una de las 8 secciones que componen al **ULN2803** puede verse en el diagrama siguiente:

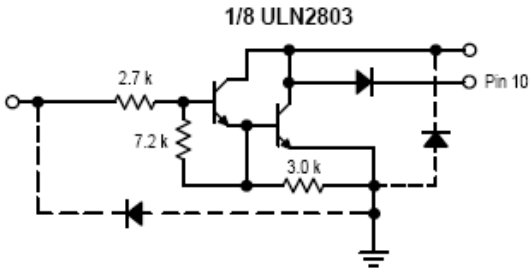


FIGURA 5.38. COMPONENTE INTERNO DEL ULN2803.

La conexión Darlington se trata de conectar dos transistores en cascada, esto es el emisor de un transistor es la base del otro como se aprecia en la figura 5.39.

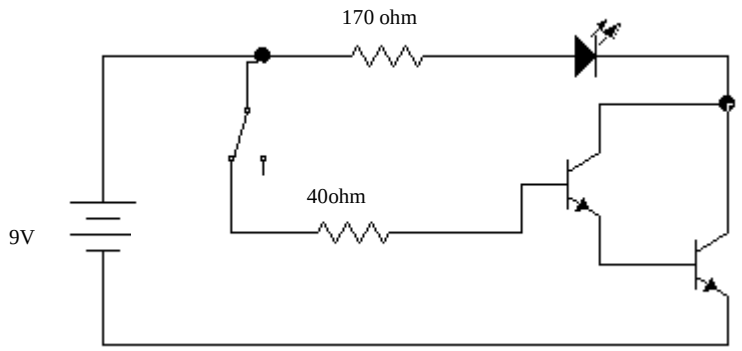


FIGURA 5.39. CONEXIÓN DARLINGTON.

Normalmente los dos transistores vienen en un encapsulado, también existen integrados con varios Darlington, y aunque la resistencia de la base es grande, el transistor conduce corriente eléctrica.

Y su función principal es proporcionar corriente necesaria para mover un motor o actuar la bobina de un relé. Permitiendo extraer 500 mA de salida.

Para ambos circuitos de las figuras 5.36 y 5.37, se utilizo una fuente de poder ATX de cualquier computadora personal para proporcionar 12 y 5 volts respectivamente y poder mantener la corriente constante en los circuitos y también en los motores, ya que el cable amarillo de la fuente ofrece teóricamente 12 V. y el rojo da 5 V.

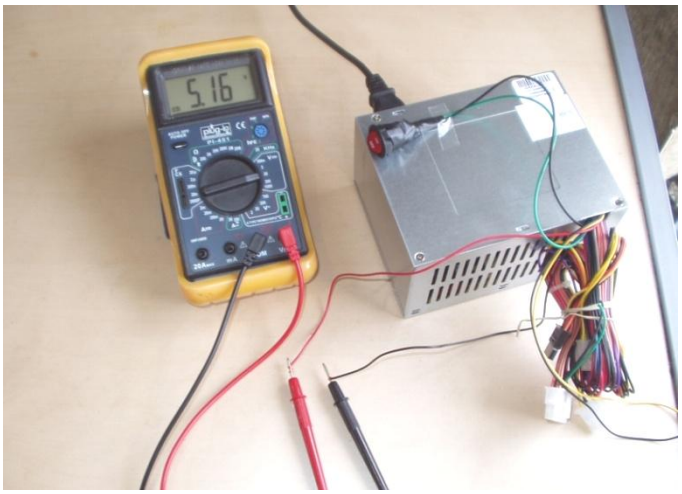


FIGURA 5.40. VALOR REAL 5.16 V.

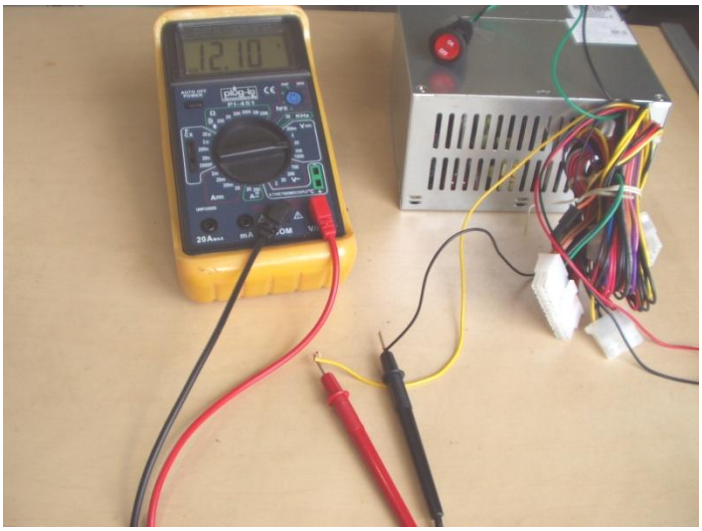


FIGURA 5.41. VALOR REAL 12.10 V.

En la fuente se realizo un puente eléctrico por medio de un interruptor entre el cable verde y negro para encender la fuente de poder, sin la necesidad de conectarla a la computadora, ya que el cable verde corresponde al "power" (es decir al encendido o apagado de la PC).

En la siguiente figura se muestra un circuito con el integrado ULN2803 con lo cual le da la potencia necesaria para los motores, a parte se muestra la alimentación a los motores cable negro y rojo (5V). Y la alimentación al circuito cable negro y amarrillo (12V)

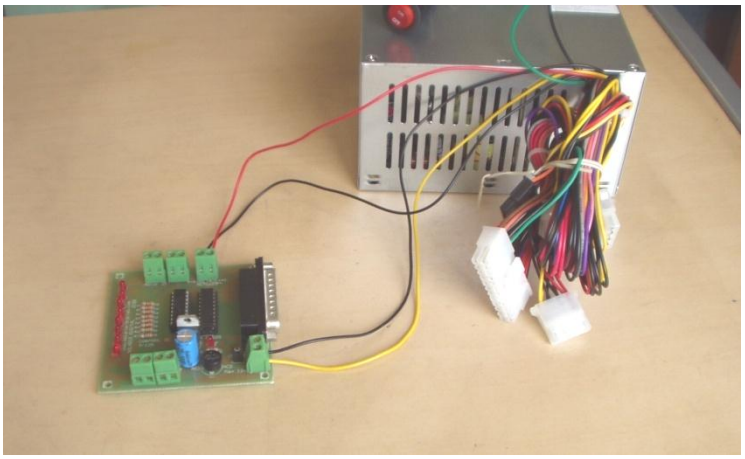


FIGURA 5.42. CONEXIÓN DE LOS CABLES RESPECTIVAMENTE.

En la figura 5.43, se ve la conexión del cable paralelo al circuito encargado de dar la potencia necesaria y del motor unipolar paso a paso el cual contiene cuatro cables y uno común que se conecta al positivo de 5V de la fuente.

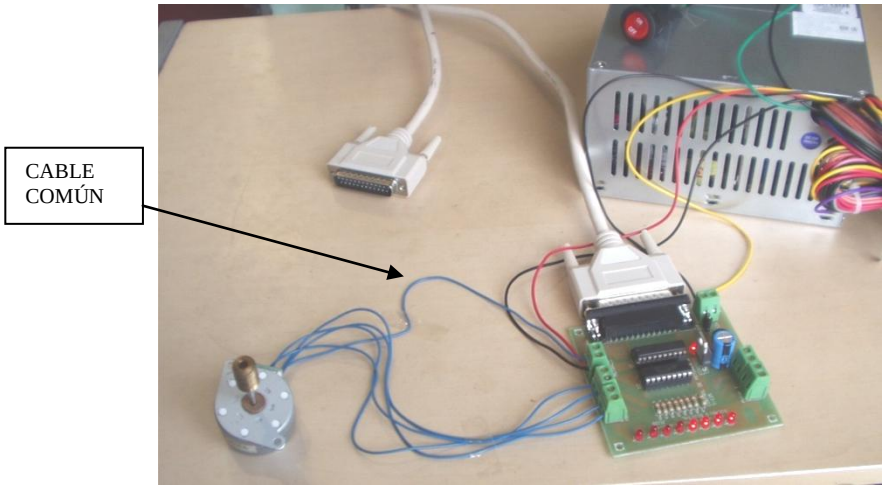


FIGURA 5.43. CONEXIÓN DEL MOTOR UNIPOLAR Y CABLE PARALELO.

Sin embargo también en Visual Basic se puede se realiza el mismo control no obstante la programación es superficial a diferencia del lenguaje C, que se trabaja de manera más directa con el hardware de la computadora.

FIGURA 5.44. INTERFAZ EN VISUAL BASIC.

CONCLUSIONES

El software de sistemas consta de programas que coordinan las diversas partes del sistema computacional para hacerlo funcionar rápida y eficazmente. Las actividades realizadas por los programas del sistema son muy diversas. Estos programas a menudo deciden dónde se almacenarán los datos que ingresan en el sistema, programan todos los trabajos que esperan su procesamiento en casi todas las grandes computadoras, llevan el registro de quién intenta ingresar en la computadora y la cantidad de tiempo que se consume en cada trabajo, y también traducen el programa de aplicación que se use al código binario que la computadora comprende.

La ingeniería en computación esta comprometida con su entorno, abierto al cambio, creativo y en permanente búsqueda de la innovación, capaz de trabajar de manera individual o coordinadamente en grupos interdisciplinarios; analizando, proponiendo e implementando soluciones a problemas en las organizaciones que involucren el desarrollo de software, interconexión de computadoras y automatización de procesos (tanto administrativos, así como productivos y de control eléctrico y electrónico), teniendo los siguientes objetivos:

- La automatización eficientemente de los procesos participando en equipos interdisciplinarios, aplicando las tecnologías de cómputo actuales e incentivando el desarrollo sustentable de la organización.

- El desarrollo de software de manera sistematizada para eficientar la comunicación y los procesos administrativos y técnicos en las organizaciones con una actitud ética y responsable.

- La aplicación y adaptar responsablemente las nuevas tecnologías de sistemas de cómputo y redes informáticas, acorde a las necesidades de las organizaciones.

Conjuntamente aplicar conocimientos sólidos en Matemáticas y Física, y con conocimientos generales de Química; así como de las áreas de sistemas de programación (software), sistemas electrónicos digitales (hardware), control y comunicaciones, que le permiten responder a las diversas necesidades que se presentan en el campo de trabajo de la Ingeniería en Computación, aplicado al control de los fenómenos eléctricos que suceden en los equipos experimentales que se encuentran en el laboratorio específico.

ANEXO

MEMORIA TIPO RIMM

Los módulos de memoria RIMM utilizan una tecnología denominada RDRAM desarrollada por Rambus Inc. a mediados de los años 90 con el fin de introducir un módulo de memoria con niveles de rendimiento muy superiores a los módulos de memoria SDRAM de 100 Mhz y 133 Mhz disponibles en aquellos años (finales de los 90's).

Los módulos RIMM RDRAM cuentan con 184 pins y debido a sus altas frecuencias de trabajo requieren de difusores de calor consistentes en una placa metálica que recubre los chips del módulo. Se basan en un bus de datos de 16 bits y están disponibles en velocidades de 300MHz (PC-600), 356 Mhz (PC-700), 400 Mhz (PC-800) y 533 Mhz (PC-1066) que por su pobre bus de 16 bits tenia un rendimiento 4 veces menor que la DDR. La RIMM de 533MHz tiene un rendimiento similar al de un módulo DDR133.

BIBLIOGRAFÍA

STOREY, NEIL
ELECTRÓNICA DE LOS SISTEMAS A LOS COMPONENTES
EDITORIAL ADDISON-WESLEY IBEROAMERICANA (1992).

ENRÍQUEZ, GILBERTO
INSTRUMENTACIÓN EN PROCESOS INDUSTRIALES
EDITORIAL LIMUSA (2004).

ROMERO, ANTONIO
COMPUTACIÓN
EDITORIAL LIMUSA (2003).

PARRA, LEOPOLDO
REPARACIÓN Y ENSAMBLADO DE COMPUTADORAS
EDITORIAL FORTON

LÓPEZ PÉREZ, BENJAMÍN
PROGRAMACIÓN EN C.
EDITORIAL SERVITEC (1998).

WILDI Y DEVITO
EXPERIMENTOS CON EQUIPOS ELÉCTRICOS
EDITORIAL LIMUSA.

MESOGRAFÍA

www.wikipedia.com.mx.

www.teleline.terra.es/personal/fremiro

www.ucontrol.com.ar

www.dictionary.zdnet.com

www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm