67
24

VNIVERMDAD NACIONAL AVPNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DIVISION DE INGENIERIA ELECTRICA, ELECTRONICA Y EN COMPUTACION

SISTEMA DE DIAGNOSTICO PARA PCs E IMPRESORAS

T E S I S

PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

(AREA ELECTRICA-ELECTRONICA)

P R E S E N T A N :

ARACELI FLORES SOTO

FRANCISCO ROMAN PONCE SALDAÑA

VICTOR JOSE LUIS RODRIGUEZ MARTINEZ

DIRECTOR DE TESIS:
ING. MIGUEL ANGEL CRUZ LEON

MEXICO, D. F. 1996

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México y muy especialmente a la Facultad de Ingeniería, Alma Mater que nos ha permitido llegar a la culminación de nuestros estudios profesionales.

A nuestros profesores, quienes tuvieron a bien transmitirnos sus conocimientos y que ha sido apoyo y ejemplo para nosotros.

Al Ing. Miguel Angel Cruz León, director del presente trabajo. Por tu apoyo, muchas gracias.

A nuestro amigo y compañero Samuel, con un especial agradecimiento por tu enorme y valiosa colaboración en la realización de este trabajo.

AGRADECIMIENTOS

La presente tesis es elaborada no solo por los autores , existe un gran apoyo hacia nosotros; por tal motivo agradezco la confianza y aliento brindado por mis padres José y Elba, y mis hermanas Esther y Estela, pues gracias a ellos tengo la oportunidad de tener una formación profesional.

Una especial dedicatoria a todos mis tios y primos que han creido que cumpliría las metas que me impuse.

Gracias a mis compañeros por permitirme contar con su amistad.

VICTOR JOSE LUIS

A mis padres, Porfirio y Josefina; a mis hermanos Clara, Rubén y Lourdes; gracias por su compañía, comprensión y por todo el apoyo que me brindaron durante mi formación como profesionista .

A mi abuelita Anita (q.e.p.d.), donde quiera que se encuentre; le agradezco los consejos y el apoyo que me dió durante el tiempo que estuvo con nosotros.

A mis tíos y primos, gracias por su compañía y por impulsarme a concluir con éxito mis estudios profesionales .

A mis compañeros, gracias por permitirme contar con su amistad y a hacer más agradables los años de estudio.

FRANCISCO ROMAN

A *mis padres Crucita y Max, porque gracias a ustedes he tenido la oportunidad de concluir mis estudios profesionales y a quienes sencillamente debo todo. Gracias por su cuidado, apoyo, paciencia, cariño y campaña . . .*

Les Quiero Mucho.

A *mis hermanos, Aurora, Bernardo y Xóchitl, quienes verdaderamente han sido mi mejor e*

También les Quiero.

A *mis dos Auroras que aún cuando lejos, nunca me dejaron sola . . .*

Nunca las Olvidaré.

C *on un especial afecto a los ingenieros Carlos Sánchez Mejía, Elvira Dávalos Paz, Ana de Gortari Pedroza y Jesús Roviroza López . . .*

Para Ustedes Todo mi Respeto.

A *mis amigos, que al ser tantos no me atrevo a nombrarlos, pero que tienen un lugar muy especial en mi; compañeros de clases y de trabajo, scouts, religiosos, militares y todos los demás . . .*

Muchas Gracias por su Compañía.

. . . no se puede pensar más, cuando no se piensa en ello.

ARACELI

TESIS

COMPLETA

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

INDICE

CAPITULO III: IMPRESORAS Y PROTOCOLOS

CAPITULO IV: ANALISIS DE RENDIMIENTO

CAPITULO V: DISEÑO E IMPLEMENTACION DE INTERFACES

APENDICES

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

INTRODUCCION

El uso y manejo de equipos de cómputo se ha convertido en la actualidad en una de las herramientas más utilizadas. Su aplicación, cuyas ventajas son bien conocidas, se ha extendido por los diferentes ámbitos del desarrollo humano.

Siendo una realidad que los equipos de cómputo son utilizados por usuarios que la mayoría de las veces no cuentan con conocimientos técnicos acerca de dichos equipos, o su uso está enfocado únicamente al manejo de algún paquete en particular, es imposible suponer que todas las personas que empleen una computadora personal puedan estar suficientemente capacitadas para atender y prevenir cualquier tipo de anomalía que pudiese presentar el sistema antes o durante su operación.

Para prevenir y atender estos aspectos se han desarrollado diferentes técnicas. Por un lado existe la asistencia directa del personal experto, que normalmente puede ser obtenida por parte del fabricante del producto o de algún otro organismo que ofrezca ese servicio. Por otro lado, el software diseñado para diagnosticar las diferentes partes de un sistema de cómputo tiene como ventaja la sencillez con que pueda ser utilizado, pues basta con ejecutar un programa para obtener un reporte del funcionamiento que se observa en la computadora; pero tiene como desventaja que las pruebas son realizadas por el programa sin intervención alguna del usuario y el informe resultante podría no ser lo suficientemente claro para deducir de él la fuente precisa de la falla o con lo que está relacionada.

La última técnica es la más efectiva. Consiste en la instalación interna de una tarjeta de diagnóstico configurada con las características del sistema en particular y que ofrece un sondeo constante de la operación; lamentablemente el costo es muy elevado y su manejo requiere un conocimiento profundo del funcionamiento de equipos de cómputo para lograr interpretar los resultados de dichos diagnósticos.

Por lo tanto, el presente trabajo propone desarrollar un módulo de diagnóstico capaz de atender la localización de fallas en computadoras personales e impresoras, con las características suficientes para poder ser empleado por personal de mantenimiento especializado o por usuarios con las inquietudes suficientes y conocimientos básicos para adentrarse en esta área.

La gama de pruebas que se abarcan está compuesta por las que se refieren a subsistemas tales como: video, unidades de disco, memoria, puertos e impresoras.

Tomando en cuenta las tendencias de marcas predominantes dentro del mercado de computadoras personales, se buscó configurar el sistema para ser utilizado en productos tales como IBM, HP y EPSON principalmente, garantizando que, en el caso de no trabajar con algunas de estas marcas, la compatibilidad sea total.

El diagnóstico se realiza en las siguientes modalidades: la primera, mediante la utilización de un programa que se ejecuta utilizando el teclado de la computadora y que efectúa paso a paso las diferentes pruebas que pueden ser elegidas de un menú en pantalla (configuración del equipo).

La segunda consiste en conectar el módulo a los puertos paralelos (utilizados principalmente por impresoras) y mediante la interacción constante entre el usuario y el módulo por medio de un programa, sea posible localizar el origen o causa de la falla.

Para diagnosticar adecuadamente alguna falla, se debe conocer internamente el funcionamiento de una computadora, por lo que se recopiló información acerca de lo que son y como funcionan los Sistemas Operativos, debido a que estos administran los diversos recursos (manejo de procesos, manipulación de interrupciones, funciones de almacenamiento, etc.) con los que cuenta una computadora y permiten la comunicación entre el usuario y ésta.

Otros elementos importantes dentro del funcionamiento de una computadora son el software y el hardware; de los cuales el primero se refiere a todos los programas, instrucciones, paquetes y lenguajes de programación que son hechos por medio de algoritmos para la resolución de problemas y/o la simplificación de trabajos; mientras que el segundo elemento se refiere a la parte física de la computadora como pueden ser: dispositivos de entrada/salida (unidades de disco, monitor, teclado, mouse, etc.), conexiones y procesadores. Por lo tanto, el software proporciona una serie de programas a ejecutar por medio del hardware, esto adapta a las computadoras de acuerdo a las necesidades del usuario (almacenar, leer e imprimir información).

Uno de los dispositivos más utilizados para desplegar información después del monitor, es la impresora, esta tiene como función obtener copias impresas en papel de los programas y/o datos que trabaja la computadora. La mayoría de las fallas que se pueden encontrar en las impresoras no son mecánicas, muchas veces se deben a una inadecuada conexión en el puerto, falla en el software o desconocimiento de las funciones incluidas en ella.

Existen diversos tipos de impresoras: matriciales, de inyección de tinta, láser, etc. De las cuales las más sencillas de analizar son las matriciales, ya que son la base de las impresoras actuales de todos los tipos.

En el caso de que se requieran características específicas de impresión (ajuste de márgenes, calidad de impresión, espacios, entre otras), se debe recurrir al uso de los protocolos (serie de instrucciones programables que permiten configurar la impresora). Para este trabajo, en el diagnóstico de impresoras se utilizaron protocolos correspondientes a impresoras en modo EPSON matricial de 9 agujas, ya que es un tipo estándar que puede ser compatible en la mayoría de las impresoras de su clase.

Si se desea conocer la eficiencia con la que un sistema de computación cumple sus objetivos, es necesario enfocarnos a su análisis de rendimiento; es decir, conocer sus tiempos de respuesta y capacidades de ejecución.

El diagnóstico del equipo de cómputo implica una revisión de las características básicas del equipo: número de puertos, número de unidades de disco, coprocesador matemático, memoria base, identificador de equipo, fecha y nombre de fabricante de BIOS (Basic Input Output System), tipo y nombre del fabricante de la tarjeta de video, etc. Para realizar este procedimiento se efectua una revisión en las áreas de ROM BIOS por medio del programa DEBUG, el cual se encuentra incluido en las utilerías de MS-DOS.

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CAPITULO I : SISTEMAS OPERATIVOS

I.1 GENERACIONES DE SISTEMAS OPERATIVOS

Un Sistema Operativo es un administrador de recursos. Los sistemas operativos, al igual que el hardware de las computadoras, han sufrido una serie de cambios revolucionarios llamados *generaciones*. Cada generación sucesiva de hardware se ha acompañado de reducciones sustanciales en los costos, tamaño, emisión de calor, consumo de energía y por incrementos notables en velocidad y capacidad de procesamiento.

En la generación cero (década de los 40's), los primeros sistemas computacionales no contaban con sistemas operativos. Los usuarios tenían completo acceso al lenguaje de la máquina. Todas las instrucciones manejaban código de máquina.

Para la primera generación (década de los 50's), los sistemas operativos fueron diseñados para hacer más fluida la transición entre trabajos. Este fue el comienzo de los *sistemas de procesamiento por lotes*, donde los trabajos se reunían por grupos o *lotes*. Cuando el trabajo estaba en ejecución, éste tenía control total de la máquina. Al terminar cada ejecución (bien o anormalmente), el control era devuelto al sistema operativo, el cual "*limpiaba*", leía e iniciaba la operación siguiente.

La característica de la segunda generación de los sistemas operativos (primera mitad de la década de los 60's), es el desarrollo de los sistemas compartidos con *multiprogramación* y los principios del *multiprocesamiento*. En los sistemas de multiprogramación, varios programas de usuarios se encuentran al mismo tiempo en el almacenamiento principal y el procesador se cambia rápidamente de un trabajo a otro. En los sistemas de multiprocesamiento se utilizan varios procesadores en un solo sistema computacional, con la finalidad de incrementar el poder de procesamiento de la máquina.

Se desarrollaron sistemas de *tiempo compartido* que operaban en *modo interactivo o conversacional con los usuarios*, y surgieron *sistemas de tiempo real*, los cuales se caracterizan por proveer una respuesta inmediata.

La tercera generación de sistemas operativos comenzó en forma efectiva en 1964, con la introducción de la familia de computadoras Sistema/360 de IBM; fueron diseñados como *sistemas para usos generales*.

Los sistemas operativos de la tercera generación eran *sistemas de modos múltiples; grandes y costosos, e* introdujeron mayor complejidad a los ambientes computacionales, ya que para lograr que uno de estos sistemas realizara una tarea simple, los usuarios debían familiarizarse con un complicado *lenguaje de control de trabajos*, a fin de poder especificar el trabajo y los recursos requeridos.

Los sistemas de la cuarta generación (de la segunda mitad de la década de los 70's a nuestros días), constituyen el estado actual de la tecnología. Con la ampliación del uso de *redes de computadoras* y del *procesamiento en línea*, los usuarios obtienen acceso a computadoras alejadas geográficamente a través de varios tipos de terminales.

El microprocesador ha hecho posible la aparición de la computadora personal, y este es uno de los desarrollos de notables consecuencias sociales más importantes de las últimas décadas.

Ha sido necesario codificar los datos personales o de gran intimidad para que, aún si los datos son expuestos, no sean de utilidad a nadie más que a los receptores adecuados.

Los *sistemas de bases de datos* han adquirido gran importancia, su función es hacer que la información se encuentre de manera controlada al alcance de aquellos que tienen derecho de acceso por medio de terminales.

I.2 DESARROLLO DE SISTEMAS OPERATIVOS

Los laboratorios de investigación de la General Motors tienen el crédito de haber sido los primeros en poner en práctica un sistema operativo durante el inicio de la década de los 50's para su IBM 701. En 1955, la General Motors y la North American Aviation, cooperaron en la producción de un sistema operativo para la IBM 704. SHARE, la organización de usuarios de IBM, fomentó las discusiones sobre los sistemas operativos y para 1957 habían sido desarrollados varios sistemas operativos domésticos para la IBM 704.

Los primeros sistemas operativos se ocupaban principalmente de la reducción del tiempo perdido en la colocación de los trabajos en la computadora (*tiempo de preparación*) y su retirada del sistema (*tiempo de descarga*); ya que mientras los trabajos eran preparados o descargados el sistema se encontraba ocioso.

Estos sistemas trataban de minimizar el tiempo muerto y suavizar las transiciones entre trabajos, logrando su objetivo al procesar los trabajos en grupos o lotes, en vez de hacerlo en forma individual.

Además, surgió el concepto de nombre de *archivos del sistema* como medio para suavizar la transición de los ensambladores y compiladores a los enlazadores y cargadores que procesaban sus salidas.

A finales de la década de los 50's, los principales distribuidores de computadoras suministraban sistemas operativos con las características siguientes:

- Procesamiento por lotes de flujo único.
- Rutinas normales de entrada/salida, para que los usuarios no se preocuparan por los detalles del código (a nivel máquina), de los procesos de entrada y salida.
- Capacidades de transición de programa a programa para reducir el tiempo perdido al iniciar un nuevo trabajo.
- Técnicas de recuperación de errores que automáticamente "hacen la limpieza" después de que un trabajo termina anormalmente, permitiendo la iniciación del siguiente con la mínima intervención del operador.
- Lenguajes de control de trabajos, que permitían a los usuarios gran parte de los detalles al definir sus trabajos y los recursos que ellos requerían.

Durante estos años los sistemas operativos solían utilizarse en sistemas grandes. Muchos de los sistemas pequeños comerciales, como la serie IBM 1400, funcionaban sin sistemas operativos. Era común que los usuarios de estos pequeños sistemas utilizaran su propio Sistema de Control de Entrada/Salida (IOCS). Este IOCS era el comienzo de los sistemas operativos tal como los conocemos hoy.

En los primeros años de la década de los 60´s, los distribuidores de equipo comenzaron a proveer sistemas operativos con capacidades mucho mayores. Algunos competidores de la época fueron:

- Bendix
- Control Data Corporation
- Honeywell
- NCR
- RCA
- Burroughs
- General Electric
- IBM
- Philco
- Sperry Univac

Los sistemas operativos de este período estaban orientados hacia lotes. La *multiprogramación* era de uso común como medio para aumentar la capacidad de ejecución del sistema, utilizando la disparidad de velocidades entre los dispositivos de entrada/salida y el procesador.

Surgieron los *sistemas de multiprocesamiento* en los que cooperaban varios procesadores, algunas veces como sistemas computacionales independientes comunicándose entre sí, y otras como procesadores múltiples compartiendo una memoria común.

La implementación y operación de reservaciones de American Airlines constituyó el primer sistema de procesamiento de transacción importante, en el cual los usuarios desde lugares remotos se comunicaban con la computadora central por medio de terminales de máquina de escribir conectadas en línea (*directamente*) con ella. Debido a esto, el enlace entre las computadoras y las comunicaciones fue el desarrollo más significativo de este período.

En 1963, Burroughs introdujo en el mercado el sistema operativo *Programa de Control Maestro (MCP)* para su sistema B5000, el cual contenía varias características comúnmente encontradas en los sistema actuales, por ejemplo:

- Multiprogramación.
- Multiprocesamiento.
- Almacenamiento virtual.
- Sistema operativo escrito en lenguaje de alto nivel.
- Capacidad de depuración del lenguaje fuente.

*** La Familia de Computadoras IBM, Sistema/360.**

En abril de 1964, IBM anunció su serie de computadoras Sistema/360, siendo este uno de los eventos importantes en la historia de los sistemas operativos.

Los usuarios de computadoras de este período cada vez que saturaban un sistema se veían forzados a moverse a un sistema más poderoso; pero la conversión entre sistemas era lenta, difícil y plagada de problemas de incompatibilidad.

Reconociendo estos problemas, IBM diseñó la serie de computadoras Sistema/360 para que fueran arquitectónicamente compatibles, usaran el mismo sistema operativo (OS/360) y ofrecieran mayor poder computacional cada vez que el usuario ascendiera en la serie. El problema era cómo hacer que los usuarios se involucraran a sí mismos con la serie 360, ya que estas máquinas eran incompatibles con otras de la misma época.

IBM resolvió el problema al proveer la más grande variedad de simuladores y emuladores de máquinas jamás ensamblados. Los simuladores y emuladores hacían que una computadora se pareciera a otra. A menudo los simuladores son muy lentos, pero de

producción hasta cierto punto económica. Los emuladores son de producción más costosa pero pueden ejecutar los viejos programas con mayor rapidez.

El anuncio del Sistema/360 provocó que los demás fabricantes regresaran a la mesa de diseño, trazando estrategias para poder competir con este gran desarrollo.

Una de estas estrategias consistía en copiar literalmente la arquitectura del sistema/360, proveer sistemas operativos similares al OS/360 y después vender estos a un precio mucho menor del que IBM pedía por los suyos. Esta estrategia fue seguida por RCA con su serie Spectra/70, pero descubrió que la tarea de duplicar el OS/360 y sus programas asociados era casi imposible; pudo duplicar el hardware, pero subestimó mucho lo costosa que sería la producción de un sistema similar al OS/360.

Otra de las estrategias del período utilizada por Burroughs y General Electric, era la de desarrollar productos que no fueran compatibles con la serie 360, pero que fueran más poderosos y de menor costo, sin embargo, el desarrollo de estos productos resultó ser una tarea mucho más formidable de lo que General Electric había anticipado, y esta compañía pronto siguió a RCA saliendo de la industria.

La esperanza que tenía IBM de conservar un solo sistema operativo para la serie 360 se vió frustrada por los diversos grados de sofisticación que existían entre sus usuarios. Los que empleaban pequeñas computadoras no deseaban sistemas operativos complejos con una serie interminable de características. Los usuarios mayores demandaban más funciones y mayores capacidades, así que, en vez de un solo sistema operativo (el OS/360) IBM entregó finalmente cuatro sistemas mayores durante la década de los 60´s:

- DOS/360	para los sistemas 360 más pequeños.
- OS/MFT	para los sistemas 360 medianos y grandes.
- OS/MTV	para los sistemas 360 grandes.
- CP-67/CMS	para el poderoso sistema 360/67 de tiempo compartido y almacenamiento virtual.

*** Sistemas de Tiempo Compartido.**

A finales de la década de los 50's y principios de la década de los 60's, los investigadores, tanto en la industria como en las universidades, desarrollaron cierto número de sistemas de tiempo compartido. Estos sistemas hacían posible que varios usuarios conversaran directamente con una computadora desde terminales.

Tal vez el sistema de tiempo compartido más significativo de la era fue el *CTSS (Sistema de Tiempo Compartido Compatible)* desarrollado en el Instituto Tecnológico de Massachusetts (MIT) por un grupo de investigación subsidiado por el gobierno de Estados Unidos, llamado proyecto MAC. El CTSS se ejecutaba en un hardware especialmente modificado de una IBM 7094.

La prueba del valor del tiempo compartido en un ambiente de desarrollo de programas surgió en el MIT, al ser utilizado el CTSS en la implementación del sistema operativo de la generación siguiente del Instituto: el *Multics (Servicio de Computación e Información Multientrelazada)*. Multics fue un esfuerzo conjunto del proyecto MAC del MIT, los laboratorios Bell y General Electric (más tarde Honeywell), basado en el concepto de almacenamiento virtual.

Muchos de estos sistemas fueron desarrollados por diversos grupos durante la segunda mitad de los años sesentas, estos incluían:

- Multics
- TSS, para el sistema 360/67 de IBM
- CP-67/CMS, también para el sistema 360/67 de IBM
- VMOS, de RCA
- KRONOS, de Control Data Corporation, para su serie 6000.

Estos proyectos ayudaron a los investigadores a obtener valiosas herramientas para el uso y administración del almacenamiento virtual.

I.2.1 SISTEMA OPERATIVO CP/M

La evolución en la computación que trajeron consigo las microcomputadoras fue posible gracias a varios avances:

- La disponibilidad de tecnología de circuitos integrados de bajo costo.
- La disponibilidad de discos flexibles altamente confiables de bajo costo.
- El constante aumento del precio de la mano de obra que demanda mejoras en la tecnología para mejorar la productividad.

CP/M (*Programa de Control para Microcomputadoras*) es un sistema operativo desarrollado originalmente para microcomputadoras de 8 bits, fue desarrollado por Gary Kildall mientras trabajaba como consejero de *Intel*, escribió el primer compilador PL/M *(Lenguaje de Programación para Microcomputadoras)*. En 1974, Kildall escribió la primera versión del sistema de archivos CP/M, que fue diseñado para apoyar a un compilador PL/M residente.

PL/M fue desarrollado en 1972 por MAA (*Microcomputer Applications Associates*, hoy Digital Research), como un lenguaje de programación de sistemas para ser utilizado con el microprocesador de 8 bits de Intel. PL/M se deriva del lenguaje de escritura de compiladores XPL, un lenguaje derivado de Algol de Burroughs y PL/1 completo.

Intel ha implementado con éxito PL/M en una variedad de microprocesadores y ha adoptado el lenguaje para el desarrollo de sistemas de software.

*** La Familia CP/M**

Microcomputer Applications Associates (MAA) decidió implementar el sistema CP/M como fuese y lo completó en 1974. CP/M se construyó como un sistema de un solo usuario con un sistema muy confiable de archivos. El acceso a archivos de disco flexible era mucho más rápido que el proporcionado por los dispositivos de cassette de aquellos tiempos.

Los diseñadores de CP/M dividieron entonces el sistema en dos partes principales que contienen los manipuladores de dispositivos de una parte invariante y una parte variante. La parte invariante contiene el sistema operativo de disco escrito en PL/M. La parte variante escrita en el lenguaje ensamblador de la computadora, contiene manipuladores de dispositivos de entrada/salida requeridos por la configuración local particular del hardware. Esta división posibilitó la conversión del CP/M en uno de los dos sistemas operativos más portables, el otro es el sistema UNIX.

Respondiendo a la diversificada base de dispositivos que tenían que ser soportados por el CP/M, *Digital Research* rediseñó completamente el sistema en 1979 para ser manipulado por tablas. Los parámetros que controlan la operación del sistema aparecen en tablas en lugar de ser "codificados rígidamente", por lo que suelen ser más fáciles de modificar.

De esta forma, el CP/M se convirtió en un sistema bastante general con gran parte de su operación definida por entradas de tabla y subrutinas de entrada/salida escritas para implementaciones locales.

*** Estructura del CP/M.**

CP/M consta de tres subsistemas principales (ver fig. 1.1):

- Procesador de mandatos de consola (CCP).
- Sistema básico de entrada/salida (BIOS).
- Sistema operativo básico de disco (BDOS).

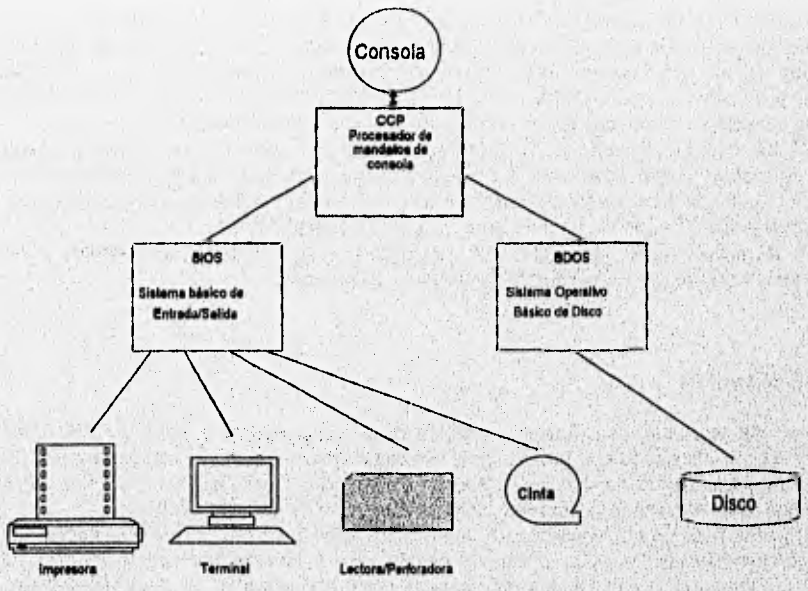


Figura 1.1 Estructura del CP/M

CCP interpreta los mandatos introducidos por el usuario y emite respuestas; libera al usuario de tener que conocer la operación interna del CP/M y del sistema computacional. CCP llama a BIOS y a BDOS para realizar el procesamiento de archivos y las operaciones de entrada/salida.

BIOS contiene los diferentes manipuladores que *envían* y *reciben datos de* los dispositivos que proporcionan información de las operaciones de entrada/salida.

BDOS contiene las diferentes rutinas de utilidad para la administración de discos. Los archivos de disco se encuentran, en general, esparcidos en pequeños bloques por toda la superficie del disco, lo que complica el almacenamiento y recuperación de información. BDOS administra estos bloques, asignando y liberando dinámicamente el almacenamiento según sea necesario.

*** Asignación de Memoria**

CP/M es un sistema de almacenamiento real. Los primeros 256 bytes (llamados Página 0) están reservados para varios parámetros del sistema. El sistema operativo reside en el almacenamiento superior. Las localidades restantes llamadas *Area del Programa Transitorio* (TPA), están disponibles para los programas del usuario. En la terminología CP/M es común referirse a la combinación de BIOS y BDOS como FDOS (Sistema Operativo Funcional de Disco). El FDOS y el CCP constituyen el CP/M (ver fig. 1.2).

Un usuario que requiera más espacio del que hay disponible en el TPA puede superponerse sobre gran parte del sistema CP/M.

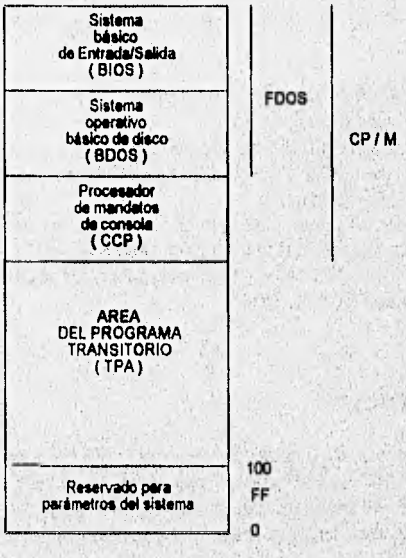


Figura 1.2 Asignación de Memoria

CP/M ve a los dispositivos como lógicos o físicos. Un dispositivo físico es aquél capaz de realizar funciones de entrada/salida. Un dispositivo lógico es un artefacto provisto por el sistema operativo para hacer más favorable la interfaz con el usuario; en realidad no existe.

Gran parte de la popularidad del CP/M se debe a la gran cantidad de usuarios de este sistema, que permite a los vendedores de software ofrecer productos sustanciales a un costo moderado y esto provoca una demanda significativa, por lo que el usuario es el primer beneficiado.

*** La Computadora Personal IBM**

UNIX fue desarrollado para minicomputadoras y está "bajando" a las microcomputadoras por medio de XENIX y otros sistemas basados en UNIX. CP/M se desarrolló para microcomputadoras de 8 bits y está "subiendo" hacia el terreno de microcomputadoras de 16 bits con sistemas como CP/M-86 (sistema uniusuario para micros de 16 bits) y MP/M-86 (sistema multiusuario para micros de 16 bits).

La computadora personal IBM anunciada en 1981, se convirtió en el centro de atención. Basada en el procesador Intel 8088 de 16 bits, adoptó un sistema operativo competitivo en esta etapa desarrollado por MicroSoft : DOS (Disk Operating System),.

I.3 ADMINISTRACION DE PROCESOS

El término *proceso* fue utilizado en primera instancia por los diseñadores del sistema Multics en los años 60's. A este concepto se le conocen diferentes definiciones, de las cuales se mencionan las siguientes:

- Un programa que se está ejecutando.
- Una actividad asincrónica.
- El "espíritu animado" del procedimiento.
- El "emplazamiento del control" de un procedimiento que se está ejecutando.
- Aquello que se manifiesta por la existencia en el sistema operativo de un "bloque de control de proceso".
- Aquella entidad a la cual son asignados los procesadores.
- La unidad "despachable".

I.3.1 ESTADOS DE PROCESO

Un proceso se encuentra en *estado de ejecución*, si utiliza en ese momento la Unidad Central de Procesos (UCP). Se dice que un proceso se encuentra *en estado de listo*, cuando podría utilizar una UCP en caso de que se encontrara alguna disponible. Un proceso se encuentra *en estado de bloqueo* si se espera que ocurra algo (por ejemplo, el fin de una entrada/salida).

Unicamente puede ejecutarse un solo proceso a la vez, pero pueden existir varios procesos listos y otros más bloqueados, por tal motivo se forma una *lista* para los procesos listos y una *lista* para los bloqueados. La primera lista tendrá mayor prioridad para que el siguiente proceso que reciba la UCP sea el primero de la misma. La lista de bloqueados se encuentra en desorden (este tipo de procesos no quedan listos) y no se desbloquean en orden prioritario; los procesos se desbloquean de acuerdo al orden en que tienen lugar los eventos que están esperando.

*** Cambios de Estado de Proceso**

Cuando una operación entra al sistema, se elabora un proceso equivalente y es insertado en la última parte de la lista de procesos listos. Cuando el proceso llega a la cabeza de la lista y la UCP se encuentre disponible, el proceso utiliza la UCP y se dice que hace un *cambio de estado*: del estado de listo al de ejecución. La asignación de la UCP al primer proceso de la lista de listos es llamado *despacho*, y es ejecutado por la entidad del sistema llamada *despachador*.

Este cambio se indica de la manera siguiente:

despacho (nombre del proceso): listo → en ejecución

Para evitar que cualquier proceso se apropie del sistema, ya sea de manera accidental o en forma premeditada, el sistema operativo ajusta un *reloj de interrupción* del hardware para permitir al usuario ejecutar su proceso durante un intervalo de tiempo específico.

Se pueden definir cuatro cambios de estado:

- Despacho (nombre del proceso): listo → en ejecución
- Tiempo excedido (nombre del proceso): en ejecución → listo
- Bloque (nombre del proceso): en ejecución → bloqueado
- Despertar (nombre del proceso): bloqueado → listo

La única transición iniciada por el propio proceso del usuario es el bloque, las demás son iniciadas por entidades ajenas al proceso.

I.3.2 BLOQUE DE CONTROL DE PROCESO

La aparición de un proceso en un sistema operativo es un *bloque de control de proceso (PCB)*. El PCB es una estructura de datos que contiene cierta información importante acerca del proceso, incluyendo:

- Estado actual del proceso.
- Identificación única del proceso.
- Prioridad del proceso.
- Apuntadores para localizar la memoria del proceso.

- Apuntadores para asignar recursos.
- Area para preservar registros.

El PCB es un almacenamiento central de información que permite al sistema operativo localizar toda la información importante del proceso.

Los sistemas que administran los procesos deben ser capaces de realizar ciertas operaciones sobre los mismos, estos incluyen:

- Crear un proceso.
- Destruir un proceso.
- Suspender un proceso.
- Reanudar (Resume) un proceso.
- Cambiar la prioridad de un proceso.
- Bloquear un proceso.
- Despertar un proceso.
- Despachar un proceso.

La creación de un proceso trae como consecuencia varias operaciones:

- Dar nombre al proceso.
- Insertar un proceso en la lista del sistema de procesos conocidos.
- Determinar la prioridad inicial del proceso.
- Crear el bloque de control de proceso.
- Asignar los recursos iniciales del proceso.

Un proceso puede *crear otro nuevo*, el que lo crea se llama *proceso padre* y el creado se llama *proceso hijo*. Esto genera una *estructura jerárquica de los mismos*.

Destruir un proceso significa borrarlo del sistema.

Un proceso suspendido no puede continuar hasta que otro lo active y se reanude la operación. Las suspensiones suelen durar un breve periodo de tiempo.

Reanudar (o activar) un proceso implica reiniciarlo en el punto donde fue suspendido.

La destrucción de un proceso es mucho más complicada. En algunos sistemas, un proceso creado se destruye automáticamente al destruir al padre, en otros, los procesos creados continúan independientemente de sus padres y la destrucción de un padre no tiene ninguna consecuencia sobre los hijos creados.

Cambiar la prioridad de un proceso sólo implica cambiar el valor de la prioridad en el bloque de control de procesos.

*** Suspensión y Reanudación**

Estas operaciones son importantes por varias razones:

- Si un sistema está funcionando deficientemente y puede fallar, entonces los procesos que están en ejecución deberán ser suspendidos para reanudarlos una vez que la falla se ha corregido.
- Un usuario que piense que algo no es correcto en un proceso, puede suspenderlo hasta que determine si éste funciona bien o mal.

- Algunos procesos pueden ser suspendidos y reanudados más tarde, cuando la carga de trabajo vuelva a sus niveles normales.
- Una suspensión puede ser iniciada por el propio proceso o por otro proceso distinto.

Un proceso en estado de listo sólo puede ser suspendido por otro, realizando el siguiente cambio:

suspende (nombre del proceso):listo → suspendido listo

Un proceso en estado de suspendido listo puede convertirse en listo por otro proceso; realizando el cambio:

reanuda (nombre del proceso):suspendido listo → listo

Un proceso en estado bloqueado puede ser suspendido por otro, para ello se realiza el cambio:

suspende (nombre del proceso):bloqueado → suspendido bloqueado

Un proceso en estado de suspendido bloqueado puede ser reanudado por otro, el cambio realizado es el siguiente:

reanuda (nombre del proceso): suspendido bloqueado → bloqueado

En general, una actividad de alta prioridad como lo es la suspensión, debe ser ejecutada inmediatamente. Cuando al fin ocurre la terminación (si es que ocurre) el proceso en estado suspendido bloqueado realiza la transición:

terminación (nombre del proceso):suspendido bloqueado → suspendido listo

I.3.3 PROCESAMIENTO DE INTERRUPCIONES

Una interrupción es un evento que altera la secuencia en que el procesador ejecuta las instrucciones; este suceso se genera por medio del hardware de la computadora. Cuando ocurre una interrupción:

- El sistema operativo obtiene el control.
- El sistema operativo guarda el estado de proceso interrumpido.

Una interrupción debe ser específicamente iniciada por un proceso en estado de ejecución o por un evento que puede estar relacionado o no a dicho proceso.

Los tipos de interrupciones que se mencionan a continuación se refieren a los procesadores IBM:

- *Interrupciones SVC* (llamada al supervisor). Una SVC es una petición generada por el usuario para un servicio particular del sistema.

- *Interrupciones de entrada/salida*. Envia señales a la UCP indicando que el estado de un canal o dispositivo ha cambiado.

- *Interrupciones externas*. Son creadas por varios eventos, como la terminación de conteo de un reloj de interrupción; la presión de la tecla de interrupción en el teclado o la recepción de una señal de otro procesador de un sistema multiprocesador.

- *Interrupciones de reinicio*. Ocurren cuando el usuario presiona el botón de reinicio o cuando llega una instrucción SIGP (procesador de señales) de otro procesador en un sistema multiprocesador.

- *Interrupciones de verificación de programa*. Son causadas por varios tipos de errores formados al ejecutar un proceso.

- *Interrupciones de verificación de la máquina*. Son causadas por un mal funcionamiento del hardware.

El sistema operativo incluye rutinas llamadas *manipuladores de interrupciones* (IH) para procesar cada tipo de interrupción. Existen seis manipuladores de interrupciones:

- IH SVC.
- IH de entrada/salida.
- IH externo.
- IH de reinicio.
- IH de verificación de programa.
- IH de verificación de la máquina.

Cuando se produce una interrupción, el sistema operativo guarda el estado del proceso interrumpido y dirige el control al manipulador de interrupciones adecuado. Esto se lleva a cabo mediante la técnica llamada *cambio de contexto*.

Las *palabras de estado de programa* (PSW) controlan el orden de ejecución de las instrucciones y contienen información diversa sobre el estado del proceso. Cuando se completa el proceso de interrupción, la UCP es enviada al proceso que estaba en ejecución en el momento de la interrupción o al proceso listo de más alta prioridad.

*** El Núcleo del Sistema Operativo**

El núcleo representa sólo una pequeña parte del código de todo el sistema operativo, pero se encuentra entre los códigos de más amplio uso. Por tal motivo, el núcleo permanece regularmente en el almacenamiento primario, en tanto que otras partes del sistema operativo son transportadas de un lado a otro entre el almacenamiento primario y el secundario, según las necesidades.

El núcleo inhabilita las interrupciones mientras responde a una de ellas; las interrupciones son habilitadas nuevamente después de completar el proceso de atención.

Un núcleo de sistema operativo contiene normalmente el código necesario para la realización de las funciones siguientes:

- Manipulación de interrupciones.
- Creación y destrucción de procesos.
- Cambio de estados de proceso.
- Despacho.

- Suspensión y reanudación de procesos.
- Sincronización de procesos.
- Comunicación entre procesos.
- Manipulación del bloque de control de proceso.
- Soporte de las actividades de entrada/salida.
- Soporte de la asignación y desasignación del almacenamiento.
- Soporte del sistema de archivos.
- Soporte de un mecanismo de llamada/regreso al procedimiento.
- Soporte de ciertas funciones contables del sistema.

I.3.4 INTERBLOQUEO

Un proceso dentro de un sistema de multiprogramación se encuentra en estado de interbloqueo si está esperando por un evento determinado que no va a ocurrir. En el interbloqueo de un sistema uno o más procesos están en una situación muy cerrada.

Cuando los recursos son compartidos entre varios usuarios, donde cada uno mantiene un control exclusivo sobre determinados recursos asignados a ese usuario, pueden producirse interbloqueos, en los cuales los procesos de algunos usuarios no podrán llegar a su término.

En cualquier sistema que mantenga procesos en espera mientras realiza la asignación de recursos y procesa la planeación de decisiones, es posible que un proceso sea aplazado indefinidamente mientras otros reciben la atención del sistema. Esta situación llamada *postergación indefinida* puede ser en ocasiones de consecuencias equivalentes a un interbloqueo.

Cuando los recursos son planeados en función de prioridades, es posible que un proceso dado espere indefinidamente en tanto continúen llegando procesos de prioridades más importantes.

En algunos sistemas, el retraso indefinido se evita al permitir que la prioridad de un proceso aumente mientras espera por un recurso, esto se llama *envejecimiento*. La prioridad de ese proceso será mayor a la de todos los que lleguen y conseguirá recibir el servicio requerido.

La teoría matemática de colas proporciona el tratamiento formal de los sistemas que administran procesos en espera.

* Recursos

Se consideran *recursos apropiativos* a la UCP y a la memoria principal.

Un programa de usuario que ha pedido una operación de entrada/salida no puede hacer uso efectivo de la memoria principal hasta que se completa dicha operación. La UCP debe ser intercambiada rápidamente (multiplexada) entre un gran número de procesos que se encuentran en espera por el servicio del sistema para mantener todos estos progresando en forma ordenada. La *apropiatividad* es muy importante para el éxito de los sistemas computacionales multiprogramados.

Ciertos recursos son *no apropiativos* y no pueden sacarse de los procesos a los que están asignados. Por ejemplo, mientras una unidad de cinta pertenezca a un proceso no puede extraerse de éste para proporcionarlo a otro.

Algunos recursos pueden ser compartidos entre varios procesos, mientras otros están dedicados a procesos individuales. La memoria principal y la UCP son compartidas entre varios procesos. Los datos y programas son recursos que deben ser controlados y asignados.

*** Condiciones para el Interbloqueo**

Coffman, Elphick y Shoshani establecieron las cuatro siguientes condiciones necesarias que deben darse para que se produzca un interbloqueo:

- Los procesos reclaman control exclusivo de los recursos que piden (condición de *exclusión mutua*).
- Los procesos mantienen los recursos que ya les han sido asignados, mientras esperan recursos adicionales (condición de *espera por*).
- Los recursos no pueden ser extraídos de los procesos que los tienen hasta su completa utilización (condición de *no apropiatividad*).
- Existe una cadena circular de procesos en la cual cada uno de ellos mantiene a uno o más recursos que son requeridos por el siguiente proceso de la cadena (condición de *espera circular*).

Havender sugirió las siguientes estrategias para evitar varias de las condiciones de interbloqueo:

- Cada proceso deberá pedir todos sus recursos requeridos de una sola vez y no podrá proceder hasta que le hayan sido asignados.
- Si a un proceso que mantiene ciertos recursos se le niega una nueva petición, éste deberá liberar sus recursos originales y en caso necesario pedirlos de nuevo junto con los recursos adicionales.
- Se impondrá la ordenación lineal de los tipos de recursos en todos los procesos, es decir, si a un proceso le han sido asignados recursos de un tipo dado, en lo sucesivo sólo podrá pedir aquellos recursos de los tipos que siguen en el ordenamiento.

*** Evitación del Interbloqueo y el Algoritmo del Banquero**

Si las condiciones necesarias para que tenga lugar un interbloqueo están en su lugar, aún es posible evitar el interbloqueo teniendo cuidado al asignar los recursos. Tal vez el algoritmo de evitación de interbloqueo más famoso es el del banquero de Dijkstra, denominado con este nombre debido a que hace referencia a un banquero que hace préstamos y recibe pagos de una fuente dada de capital.

Los procesos reclaman uso exclusivo de los recursos que se requieren. Se permite mantener a los procesos mientras piden y esperan por otros adicionales, y no pueden apropiarse de un proceso que mantenga esos recursos. Los usuarios alivian al sistema al pedir un recurso a la vez. El sistema puede conceder o negar cada una de las peticiones; si se niega una petición, ese usuario toma todo recurso que tenga asignado y espera durante un tiempo finito hasta que le sea atendida la petición. *El sistema concede peticiones que den*

como resultado sólo estados seguros. Una petición de un usuario que dé como resultado un estado inseguro es negada hasta que pueda ser satisfecha.

El algoritmo del banquero permite proseguir a los trabajos que en una situación de prevención de interbloqueo tendrían que esperar. Pero el algoritmo contiene un número de debilidades importantes que pueden hacer que un diseñador escoja otro enfoque para el problema del interbloqueo, por lo que requiere que:

- Exista un número fijo de recursos asignables.
- La población de usuarios se mantenga constante.
- El banquero garantice que todas las peticiones serán concedidas dentro de un intervalo de tiempo finito.
- Los trabajos garanticen que los recursos van a ser devueltos dentro de un intervalo de tiempo finito.
- Los usuarios indiquen sus necesidades máximas por adelantado.

La detección del interbloqueo es el proceso de determinar si de hecho existe o no un interbloqueo, e identificar cuáles son los procesos y recursos implicados en él.

Una vez bloqueado el sistema, hay que romper el interbloqueo retirando una o más de las condiciones necesarias. Son varios los factores que dificultan la recuperación del interbloqueo:

- Puede no estar claro si el sistema se ha bloqueado o no.
- Muchos sistemas tienen medios pobres para suspender un proceso por tiempo indefinido y reanudarlo más tarde.
- Aunque existieran medios efectivos de suspensión/reanudación, seguramente estos implicarán una sobrecarga considerable y pueden requerir la atención de un operador altamente calificado.
- Recuperarse de un interbloqueo de proporciones modestas puede suponer una cantidad razonable de trabajo; un interbloqueo a gran escala puede requerir una cantidad enorme de trabajo.

En sistemas que se encuentren en operación, la recuperación suele realizarse al retirar forzosamente de él un proceso y reclamar sus recursos. Los procesos pueden ser retirados de acuerdo a un orden de prioridades:

- Las prioridades de los procesos bloqueados pueden no existir, así es que el operador deberá tomar una decisión arbitraria.
- Las prioridades pueden ser incorrectas o confusas debido a consideraciones especiales, como en la planificación a plazo fijo, en la cual algunos procesos de prioridad relativamente baja tienen una prioridad momentáneamente alta a causa de una fecha tope inminente.
- La decisión óptima de cuáles procesos retirar puede requerir de un esfuerzo considerable para determinarla.

El enfoque más deseable a la recuperación del interbloqueo está en un mecanismo efectivo de suspensión/reanudación. Esto podrá tener contenciones temporales a los procesos y luego reanudarlos sin pérdida del trabajo.

I.4 ALMACENAMIENTO PRINCIPAL

* Organización del Almacenamiento Real

Se refiere a la forma de considerar el almacenamiento de información en la computadora, buscando la manera de dar un servicio eficiente a los usuarios.

* Administración del Almacenamiento Real

Las estrategias de administración del almacenamiento no determinan el rendimiento de una organización del almacenamiento con varias estrategias. Pero es necesario no perder de vista que lo más importante siempre será dar un servicio eficiente para obtener un óptimo rendimiento.

* Jerarquía del Almacenamiento

En las décadas de los 50's y 60's, el almacenamiento principal (que generalmente era una memoria de núcleo magnético), era muy costoso pero indispensable para que los programas y los datos pudieran ser ejecutados, por ello era necesario determinar con especial cuidado la cantidad de memoria principal que debía colocarse en cada sistema computacional.

Los programas y datos que no se necesitan de manera inmediata podían permanecer en memorias secundarias tales como cintas o discos, mismos que además de ser menos costosos, contaban con una mayor capacidad de almacenamiento. Sin embargo, el almacenamiento principal tiene una ventaja determinante ante el almacenamiento secundario: su acceso es mucho más rápido.

Durante la década de los 60's se observó que debían existir cambios en la jerarquía de almacenamiento, lo cuál implicaría mejoras en el servicio. Como consecuencia se realizó el llamado *"caché"*, que es un almacenamiento de alta velocidad que supera incluso al almacenamiento principal; sin embargo, éste resultó más costoso, razón por la cual se utilizaban cachés muy pequeños.

El almacenamiento caché impone al sistema un nivel más de traspaso. Los programas del almacenamiento principal son traspasados al caché antes de su ejecución, dentro de éste son ejecutados con mayor rapidez que si se encontraran en la memoria principal (Ver Fig. 1.3).

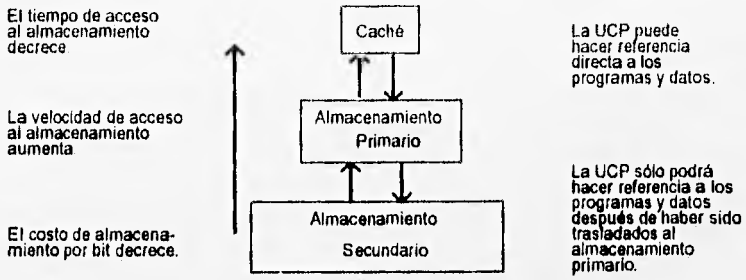


Figura 1.3 Jerarquía del Almacenamiento

I.4.I ESTRATEGIAS DE ALMACENAMIENTO

Las estrategias de almacenamiento se refieren al mejor uso de los recursos del almacenamiento principal.

Estas estrategias están divididas en categorías de la siguiente manera:

1) De búsqueda.
   a) Por demanda.
   b) Anticipada.

2) De colocación.

3) De reposición.

I.4.2 ORGANIZACION DEL ALMACENAMIENTO VIRTUAL

El término *almacenamiento virtual*, suele asociarse con la capacidad de direccionar un espacio de almacenamiento mucho mayor que el disponible en el almacenamiento primario de determinado sistema de computación. Apareció por vez primera en el sistema computacional Atlas, construido por la Universidad de Manchester, Inglaterra, en 1960.

Los dos métodos más comunes de implementación del almacenamiento virtual son la paginación y la segmentación.

* Conceptos Básicos

La clave del concepto de almacenamiento virtual, está en la separación de las direcciones a las que hace referencia un programa de las direcciones disponibles en el almacenamiento primario. Las direcciones virtuales deben ser transformadas dentro de las direcciones reales, mientras el proceso está en ejecución (Ver Figura 1.4).

Los mecanismos de traducción dinámica de direcciones (DAT) convierten las direcciones virtuales en reales al ejecutarse el proceso. Estos mecanismos (DAT) deben mantener mapas que ilustren qué direcciones del almacenamiento virtual se encuentran en el almacenamiento real y dónde están. Es preciso un método para reducir la cantidad de información del mapa para que valga la pena realizar la implementación del almacenamiento virtual. Así pues, la información se agrupa en bloques; cuando los bloques son del mismo tamaño reciben el nombre de *páginas*, y la organización del almacenamiento virtual asociada se denomina *paginación*. Cuando los bloques pueden tener diferentes tamaños se llaman *segmentos* y la organización de almacenamiento virtual asociada se conoce como *segmentación*.

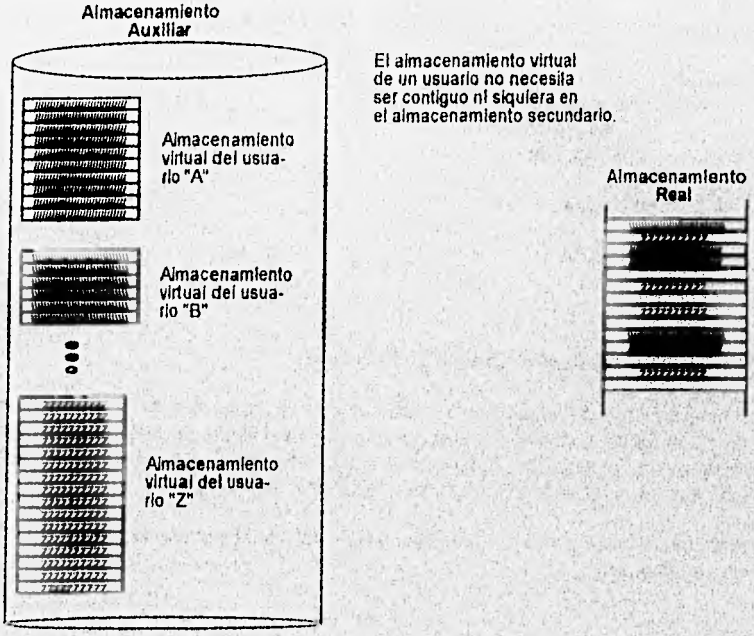


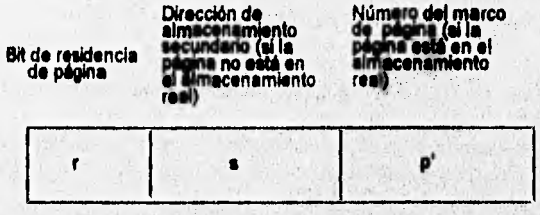
Figura 1.4 Almacenamiento Virtual

* Paginación

Una dirección virtual en un sistema de paginación es un par ordenado (p,d), donde *p* es el número de la página en el almacenamiento virtual en que reside el elemento al que se está haciendo referencia y *d* es el desplazamiento en la página p en la cual está localizado el elemento referenciado.

Un proceso puede ejecutarse si su página actual está en el almacenamiento primario.

Las páginas se transfieren del almacenamiento secundario al primario en bloques llamados *marcos de páginas*, los cuales tienen el mismo tamaño de las páginas.



r = 0 si la página no está en el almacenamiento real
r = 1 si la página está en el almacenamiento real

Los tipos de traducción de direcciones bajo la paginación proceden de la siguiente forma:

a) Traducción de direcciones de paginación por transformación directa.

Un proceso en ejecución hace referencia a la dirección virtual *v = (p,d)*. Antes que un proceso comience su ejecución, el sistema operativo carga la dirección de almacenamiento primario de la tabla de mapa de páginas en el registro origen de la tabla de mapa de páginas. La dirección base *b*, de la tabla de mapa de páginas se añade al número de página *p* para formar la dirección en el almacenamiento primario, *b+p*, de la entrada en la tabla de mapa de páginas para la página *p*. Esta entrada indica que el marco de página *p'* corresponde a la página virtual. Entonces se concatena con el desplazamiento *d*, para formar la dirección real *r* (Ver Fig. 1.5).

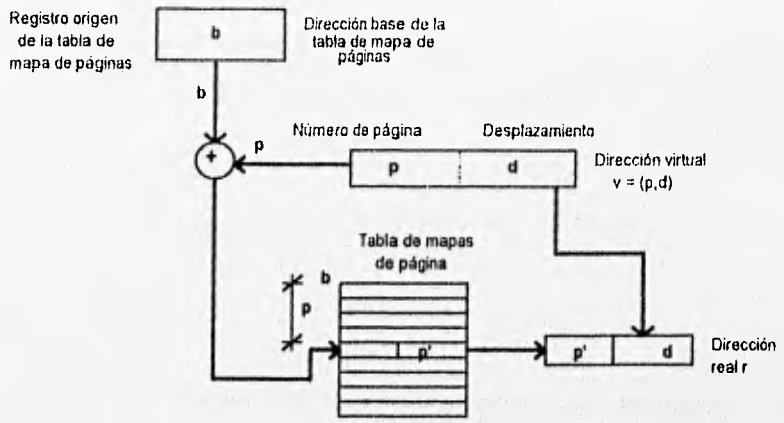


Figura 1.5 Traducción de direcciones de paginación por transformación directa

La traducción de direcciones de páginas por transformación directa puede ocasionar que el sistema de computación ejecute los programas cerca de la mitad de su velocidad normal.

b) Traducción de direcciones de paginación por transformación asociativa.

Una forma de acelerar la traducción dinámica de páginas consiste en colocar la table completa de mapa de páginas en un almacenamiento asociativo cuyo ciclo ocupa un tiempo menor que el del almacenamiento primario.

Un programa en ejecución hace referencia a la dirección virtual *v = (p,d)*. Cada entrada en el almacenamiento asociativo se busca de forma simultánea para la página *p*. Devuelve *p'* como el marco de página correspondiente a la página *p*, y *p'* se concatena con *d*, formando la dirección real *r* (Ver Fig. 1.6).

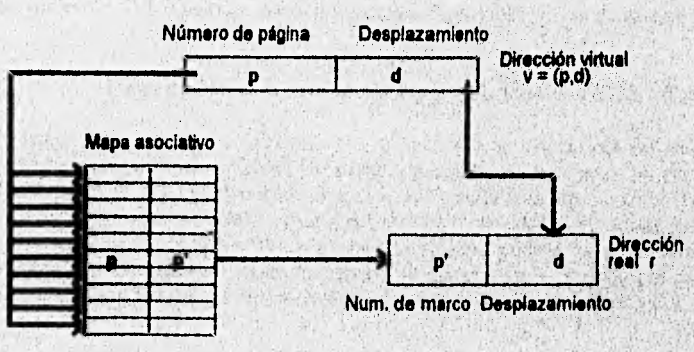


Figura 1.6 Traducción de direcciones de paginación por transformación asociativa

En el diagrama anterior se observa que las flechas del mapa asociativo penetran dentro de cada una de las células del mapa, lo cual indica que cada una de las células del almacenamiento asociativo se registra de forma simultánea para buscar una correspondencia con *p*. Esto es lo que hace costoso al almacenamiento asociativo.

c) Traducción de direcciones de paginación por combinación de transformación asociativa/directa.

Un programa hace referencia a la dirección virtual *v = (p,d)*. El mecanismo de traducción de direcciones intenta primero encontrar la página *p* en el mapa de página asociativo parcial. Si *p* se encuentra ahí entonces el mapa asociativo devuelve *p'* como el número de marco correspondiente a la página virtual *p*, y *p'* se concatena con el desplazamiento *d*, para formar la dirección real *r* que corresponde a la dirección virtual *v = (p,d)*.

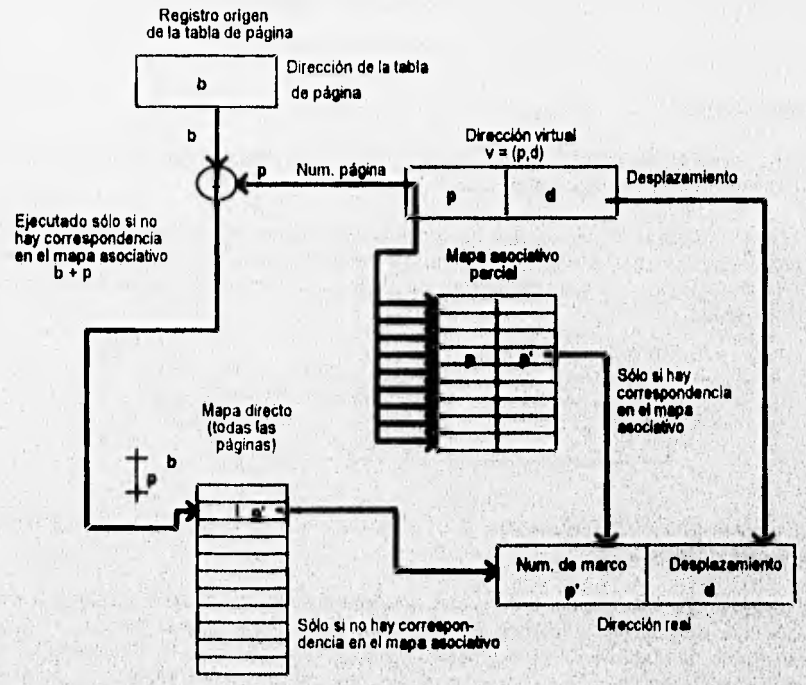


Figura 1.7 Traducción de direcciones de paginación por combinación de transformación asociativa/directa

Si no existe correspondencia para la página *p* dentro del mapa de páginas asociativo, se utiliza un mapa directo convencional. La dirección *b*, del registro origen de la tabla de mapa de páginas se añade a *p* para localizar la entrada apropiada a la página *p* en la tabla de mapa de páginas de transformación directa del almacenamiento primario. La tabla indica que *p´* es el marco de página correspondiente a la página virtual *p*, y *p´* se concatena con el desplazamiento *d* para formar la dirección real *r*, correspondiente a la dirección virtual *v = (p,d)*. Este tipo de traducción se muestra en la figura 1.7.

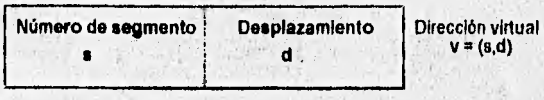
d) Compartimiento de recursos en sistemas de paginación.

Indica la necesidad de identificar cada página como compartible o no. Al hacer que las entradas de tabla de página de varios procesos apunten al mismo marco, ese marco de página es compartido por esos procesos. El compartimiento reduce la cantidad de almacenamiento primario necesario para la ejecución eficaz de un grupo de procesos y pueda ser posible que un sistema determinado mantenga mayor cantidad de usuarios.

*** Segmentación**

En los sistemas de segmentación, un programa (y sus datos) pueden ocupar varios bloques separados del almacenamiento real.

Una dirección virtual de un sistema de segmentación es un par ordenado de la manera *v = (s,d)*, donde *s* es el número del segmento del almacenamiento virtual en el cual residen los elementos referidos, y *d* es el desplazamiento en el segmento *s* en el cual se encuentra el elemento referido.



Un proceso sólo puede ejecutarse si su segmento actual (como mínimo) está en el almacenamiento primario.

La traducción dinámica de direcciones bajo la segmentación procede de la siguiente forma: un proceso en ejecución hace referencia a una dirección del almacenamiento virtual *v = (s,d)*. Un mecanismo de planificación de segmentos busca al segmento *s* en la tabla de mapa de segmentos y determina si éste se encuentra dentro del almacenamiento real comenzando en la posición *s´*. La dirección del almacenamiento real correspondiente a la dirección virtual *v = (s,d)* se forma entonces añadiendo *s´* a *d* (Ver Fig. 1.8).

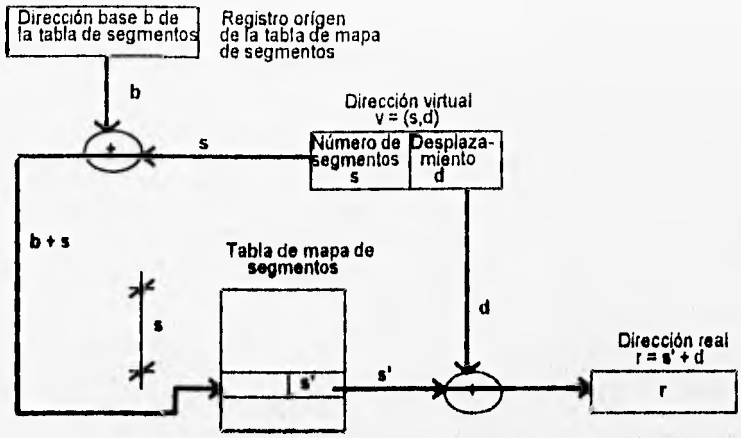


Figura 1.8 Traducción dinámica de direcciones bajo segmentación

a) Control de acceso en sistemas de segmentación.

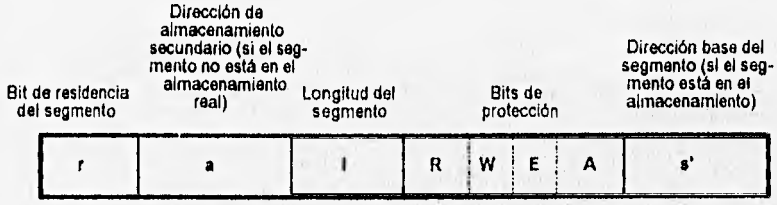
Los tipos más comunes de control de acceso usados en los sistemas actuales son: lectura, escritura, ejecución y adición. En un sistema con estos cuatro tipos de control de acceso, permitiendo o negando cada uno de ellos, pueden crearse 16 diferentes modos de control de acceso, de los cuales los más sencillos son los siguientes:

	Lectura	Escritura	Ejecución	Explicación	Aplicación
Modo 0	No	No	No	No hay permiso de acceso	Seguridad
Modo 1	No	No	Si	sólo ejecución	Un programa disponible a los usuarios, que no pueden copiarlo ni modificarlo, pero sí ejecutarlo
Modo 2	No	Si	No	Sólo escritura	Estas posibilidades no son útiles, no tiene sentido conceder acceso de la escritura si se niega el acceso a la lectura
Modo 3	No	Si	Si	Escritura/ejecución, pero no lectura	
Modo 4	Si	No	No	Sólo lectura	Recuperación de información
Modo 5	Si	No	Si	Lectura/ejecución	Un programa puede ser copiado o ejecutado, pero no modificado
Modo 6	Si	Si	No	Lectura/escritura, pero no ejecución	Protege los datos contra un intento erróneo de ejecutarlos
Modo 7	Si	Si	Si	Acceso no limitado	Este acceso se concede a los usuarios de confianza

Tabla 1.1 Control de acceso en sistemas de segmentación

b) Traducción de direcciones de segmentación por transformación directa.

Un proceso en ejecución hace referencia a la dirección virtual *v = (s,d)*. El segmento número *s* se añade a la dirección base *b*, en el registro origen de la tabla de mapa de segmentos para formar la dirección del almacenamiento real, *b+s*, de la entrada para el segmento *s* de la tabla de mapa de segmentos. Esta tabla contiene la dirección del almacenamiento primario *s'*, donde comienza el segmento. El desplazamiento *d*, se añade a *s'* para formar la dirección real, *r = d+s'*, correspondiente a la dirección virtual *v = (s,d)*.



r = 0 si el segmento no está en el almacenamiento primario
r = 1 si el segmento está en el almacenamiento primario

Bits de protección: (1 = si, 0 = no)

R = acceso de lectura
W = acceso de escritura
E = acceso de ejecución
A = acceso de adición

c) Compartimiento en un sistema de segmentación.

Una de las ventajas de la segmentación sobre la paginación es que se trata más de un concepto lógico que físico. Los segmentos no están restringidos a un tamaño fijo, ya que se les permite (dentro de límites razonables) tener el tamaño que necesiten (Ver Fig. 1.9).

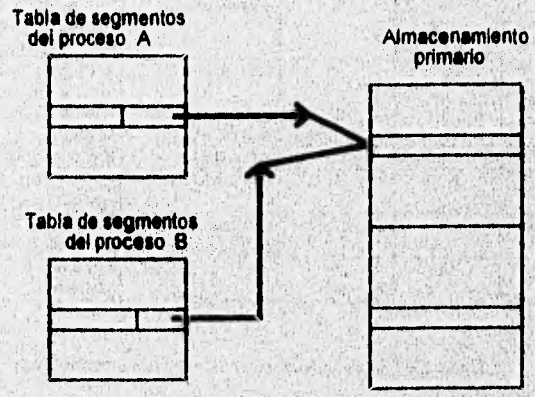


Figura 1.9 Compartimiento en un sistema de segmentación

* Sistemas de paginación/segmentación

Estos sistemas ofrecen las ventajas de las dos técnicas de organización del almacenamiento virtual. En general, el tamaño de los segmentos es múltiplo del tamaño de las páginas; no es necesario que todas las páginas de un segmento se encuentren al mismo tiempo en el almacenamiento primario y las páginas de almacenamiento virtual que son contiguas en este almacenamiento no necesitan ser contiguas en el almacenamiento real. El direccionamiento es tridimensional con una dirección de almacenamiento virtual *v*, siendo un trío ordenado *v = (s,p,d)*, donde *s* es el número de segmento, *p* es el número de página y *d* el desplazamiento en la página donde se encuentra asignado el elemento deseado.

Número de segmento	Número de página	Desplazamiento	Dirección virtual v = (s,p,d)
s	p	d	

a)Traducción dinámica de direcciones en sistemas de paginación/segmentación.

Un proceso en ejecución hace referencia a la dirección virtual. Las páginas de referencia más reciente tienen entradas en un almacenamiento asociativo.

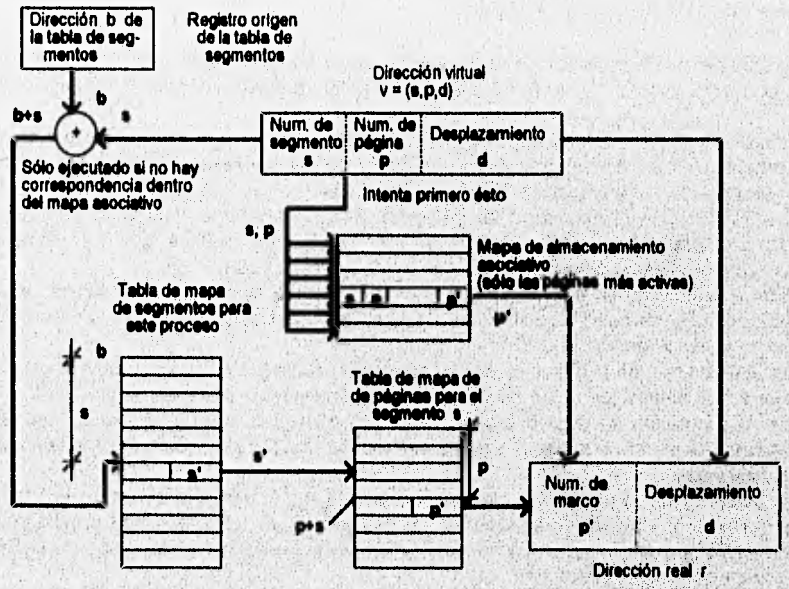


Figura 1.10 Traducción dinámica de direcciones en sistemas de paginación/segmentación

Se realiza una búsqueda asociativa para intentar localizar *(s,p)* en el almacenamiento asociativo. Si se encuentra *(s,p)*, entonces el marco de página *p'* en el cual reside dicha página en el almacenamiento primario se concatena al desplazamiento *d* para formar la dirección de almacenamiento real *r*, que corresponde a la dirección virtual *v = (s,p,d)* y se completa la traducción de dirección (Ver Fig. 1.10).

I.4.3 ADMINISTRACION DEL ALMACENAMIENTO VIRTUAL

*** Estrategias del Almacenamiento Virtual**

- *Estrategias de Búsqueda:* Tratan de los casos en que una página o segmento deben ser traídos del almacenamiento secundario al primario.
- *Estrategias de Colocación:* Tratan del lugar del almacenamiento primario donde se colocará una nueva página o segmento.
- *Estrategias de Reposición:* Tratan de la decisión de cual página o segmento se desplazará para hacer sitio a uno nuevo cuando el almacenamiento primario está completamente comprometido.

*** Estrategias de Reposición de Página**

- El principio de optimización indica que para obtener un rendimiento óptimo, la página que se va a reponer es una que no se va a utilizar en el futuro durante el periodo de tiempo más largo.
- Si buscamos una estrategia de reposición de página con poca sobrecarga y que no sea discriminatoria, es escoger al azar la página que va a ser reemplazada. Todas las páginas deben tener la misma probabilidad de ser reemplazadas.
- En la reposición de página por el sistema de primero en entrar-primero en salir (FIFO); cuando una página necesita ser reemplazada, escogemos aquella que ha tenido su oportunidad y es tiempo de que se le otorgue la oportunidad a otra.

  Con la reposición de página FIFO, ciertos patrones de referencias de páginas causan más fallos de páginas cuando se aumenta el número de marcos asignados a un proceso. Este fenómeno se llama *"anomalía FIFO"*.
- En la reposición de página menos-recientemente-usada (LRU), es seleccionada para ser reemplazada la página que no ha sido usada durante el mayor periodo de tiempo.
- En la reposición de página menos-frecuentemente-usada (LFU), la página que será reemplazada es aquella que ha sido usada con menos frecuencia o que ha sido referida con menos intensidad.
- En la reposición de página no usada-recientemente (NUR), las páginas que no han tenido uso reciente tienen poca probabilidad de ser usadas en un futuro próximo y pueden ser reemplazadas por otras nuevas. La estrategia NUR se implementa con la adición de dos bits de hardware por página, estos son:

  a) Bit referenciado = 0 si la página no ha sido referenciada
  = 1 si la página ha sido referenciada

b) Bit modificado = 0 si la página no ha sido modificada
= 1 si la página ha sido modificada

El esquema NUR da como resultado la existencia de cuatro grupos de páginas:

Grupo 1	no referenciado	no modificado
Grupo 2	no referenciado	modificado
Grupo 3	referenciado	no modificado
Grupo 4	referenciado	modificado

Las páginas de los primeros dos grupos se verán reemplazadas en primer lugar y las de los grupos 3 y 4 serán las últimas.

*** Conjuntos de Trabajo**

Un conjunto de trabajo es una colección de páginas a las cuales un proceso hace referencia.

El conjunto de trabajo de páginas de un proceso *W (t,w)*, en el momento *t*, es el conjunto de páginas referidas por un proceso durante el intervalo de tiempo del proceso *t-w* a *t*. La variable *w* se denomina *tamaño de la ventana del conjunto de trabajo*. El verdadero conjunto de trabajo de un proceso, es el conjunto de páginas que deben estar en el almacenamiento primario para la ejecución eficaz de este proceso. Los conjuntos de trabajo son transitorios y el siguiente conjunto de trabajo del proceso puede diferir sustancialmente de su conjunto de trabajo anterior.

*** Paginación por demanda**

La paginación por demanda solamente utiliza las páginas que realmente necesite el proceso y que son llevadas al almacenamiento principal.

La paginación por demanda no se encuentra libre de problemas, esto se puede ejemplificar con el siguiente diagrama (Fig. 1.11):

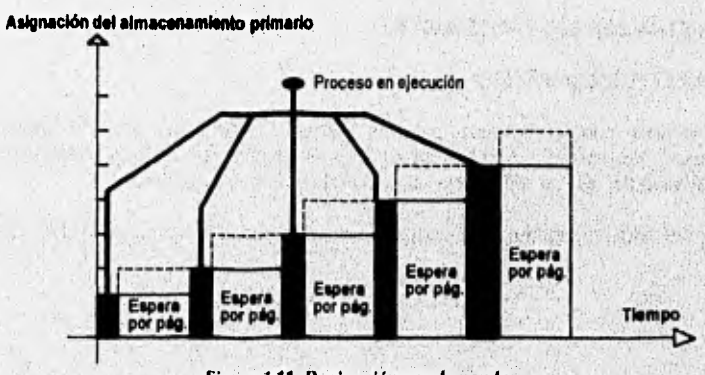


Figura 1.11 Paginación por demanda

El proceso que se esté realizando debe acumular páginas una a una. Cada vez que se referencia una nueva página el proceso debe esperar a que se transfiera al almacenamiento primario. Dependiendo de cuantas páginas de este proceso se encuentren ya en el almacenamiento primario, las esperas se hacen cada vez más costosas al ocuparse cantidades más grandes de almacenamiento por procesos en espera.

*** Paginación anticipada**

Al utilizar esta técnica el sistema operativo trata de predecir las páginas que un proceso necesitará y posteriormente carga estas páginas si existe espacio disponible.

Con la paginación anticipada, se toman decisiones correctas en la gran mayoría de los casos, por lo que el tiempo de ejecución de un proceso se reduce considerablemente.

*** Tamaño de página**

- Cuanto más pequeño sea el tamaño de una página, más páginas y marcos de las mismas habrá, lo que implica tener mayores tablas de página.

- Con páginas grandes, cantidades similares de información que nunca llegarían a ser referenciadas se paginarán hacia el almacenamiento primario.

- Un tamaño más pequeño de página ayudaría a un programa a establecer un conjunto de trabajo más cerrado, es decir, las páginas del conjunto de trabajo mantenidas en el almacenamiento real contendrán elementos referenciados con más intensidad.

- La liberación voluntaria de página puede eliminar el desperdicio y acelerar la ejecución del programa.

I.5 ADMINISTRACION DEL PROCESADOR

I.5.1 MULTIPROCESAMIENTO

Los sistemas computacionales actuales deben contar con diversas utilerías que proporcionen a los usuarios un campo de acción más amplio, esto es, que se tenga acceso al *multiprocesamiento*, que se refiere a la utilización de varios procesadores.

Esquemáticamente podemos representar a un multiprocesador como sigue (Fig. 1.12):

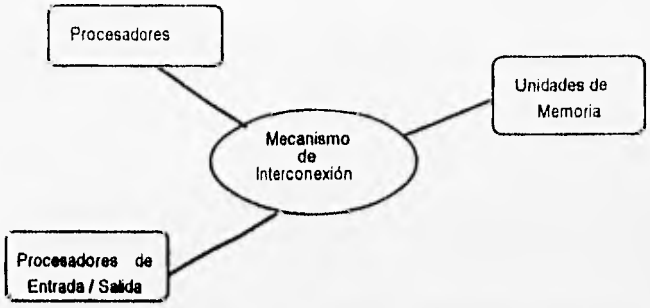


Figura 1.12 Multiprocesador

Los sistemas de multiprocesamiento deben cumplir con ciertas condiciones para que su funcionamiento sea óptimo, pero una de las más importantes es que el sistema debe ser estrictamente confiable para el usuario. Si dentro del sistema de multiprocesamiento uno de los procesadores falla, éste debe ser capaz de informar a los demás lo que ocurre para que ellos se hagan cargo del trabajo.

Uno de los factores más importantes en el funcionamiento de los sistemas de multiprocesamiento es que la confiabilidad de estos se incrementa cuando al faltar alguno de los procesadores los demás entran a sustituirlo.

El objetivo principal de los sistemas de multiprocesamiento es el incremento de la capacidad de ejecución.

Una forma de lograr lo anterior, se da al utilizar un lenguaje que permita el paralelismo en la programación y procesamiento de información. El multiprocesamiento es lo que más se acerca a esta idea; sin embargo, es necesario la utilización de varios procesadores que en la mayoría de la veces resultan ser insuficientes.

Una propuesta para lograr una explotación del paralelismo masivo es la utilización de un número de procesadores que garantice que todas las operaciones posibles a ser ejecutadas en paralelo puedan ser asignadas a procesadores separados.

Los sistemas de multiprocesamiento pueden lograr un procesamiento en paralelo. El paralelismo dentro de los programas puede ser explícito ó implícito. El paralelismo de forma explícita puede ser indicado utilizando una construcción de recurrencia. El sistema de multiprocesamiento tiene la capacidad de utilizar procesadores separados para ejecutar cada una de los proposiciones de la concurrencia y con ello lograr que el proceso termine rápidamente. Pero esto también tiene sus problemas y el principal de ellos es que un programa que contenga mucho paralelismo explícito puede introducir fácilmente errores que son difíciles de detectar.

La solución al problema anterior puede ser el uso del paralelismo implícito, el cual no se encuentra dentro del cuerpo del programa a simple vista, sino que es parte del mismo esquema del algoritmo planteado.

Existe dos técnicas que utilizan los compiladores para la explotación del paralelismo implícito y son:

- *Distribución de Ciclos:* Consiste en una estructura que implica la repetición de proposiciones hasta que ocurre una condición que marca la terminación del ciclo, por ejemplo:

```
FOR I = 1 TO 4 DO
A(I) = B(I) + C(I)
```

- *Reducción de la Altura del Arbol:* En esta técnica se utilizan las propiedades aritméticas de asociatividad, conmutatividad y distributividad. Con ello se logra producir un código que indique al sistema de multiprocesadores que realice simultáneamente algunas de las operaciones.

No todos los procesadores responden a las mismas reglas de procesamiento pero generalmente respetan las siguientes consideraciones:

- Realizar primero las operaciones que se encuentran en paréntesis anidados.
- Después realizar las operaciones que se encuentren sólo entre paréntesis sencillos.
- Realizar después las operaciones exponenciales.
- Continuar con las divisiones y multiplicaciones.
- Concluir con las sumas y/o restas.
- También debe considerarse que, si las operaciones a realizarse tienen la misma jerarquía, entonces se dará atención de acuerdo al orden encontrado de izquierda a derecha.

En la reducción de la altura del árbol utilizando las diferentes propiedades aritméticas, puede representarse como sigue (Figs. 1.13, 1.14 y 1.15):

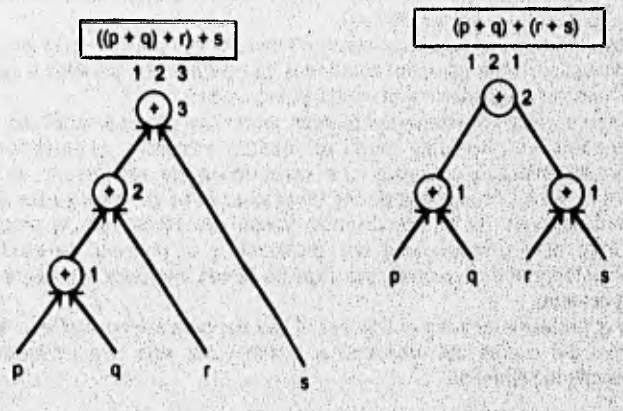


Figura 1.13 Reducción de la altura del árbol por Asociatividad

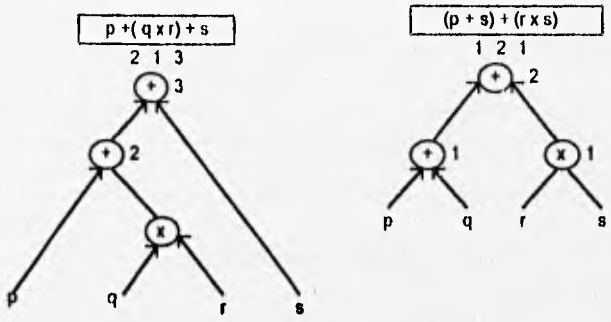


Figura 1.14 Reducción de la altura del árbol por Conmutatividad

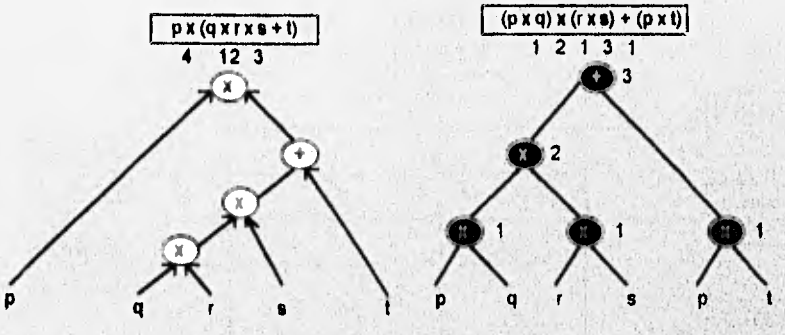


Figura 1.15 Reducción de la altura del árbol por Distributividad

*** Organización del Hardware del Multiprocesador**

Uno de los problemas con que se encuentran los sistemas de multiprocesamiento es el hecho de determinar los medios de conexión de los procesadores múltiples y los procesadores de entrada/salida a las unidades de almacenamiento.

Esencialmente existen tres organizaciones de sistemas de multiprocesadores que son:

- *Tiempo compartido* o *bus común*. Este tipo de organización utiliza un sólo camino de comunicación entre todas las unidades funcionales, como se muestra esquemáticamente en la figura 1.16:

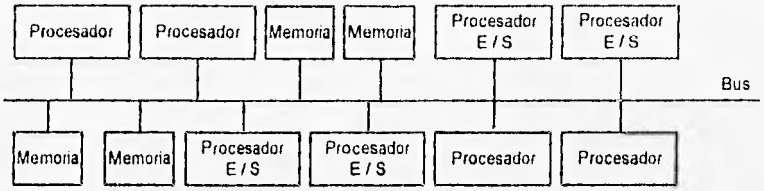


Figura 1.16 Tiempo compartido o bus común

- *Matriz de barras cruzadas e interruptores.* Este tipo de organización se crea al aumentar el número de conductores de bus común al mismo número de unidades de almacenamiento. Aquí existe un camino diferente para cada unidad de almacenamiento (Ver Fig. 1.17).

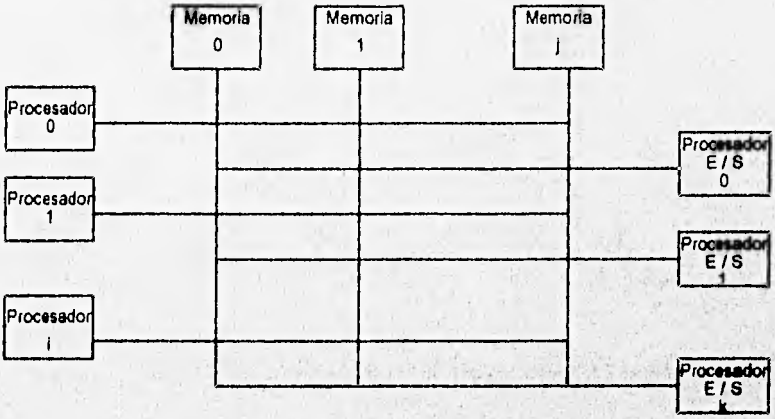


Figura 1.17 Matriz de barras cruzadas e interruptores

- *Almacenamiento de interconexión múltiple.* Este tipo de almacenamiento se lleva a cabo cuando se sacan del interruptor de barras cruzadas las lógicas de control, arbitraje de prioridades y conmutación. Aquí, cada unidad funcional puede acceder a cada unidad de almacenamiento pero sólo en una unidad de almacenamiento específica. Se suministra una conexión de almacenamiento por unidad funcional (Ver Fig. 1.18).

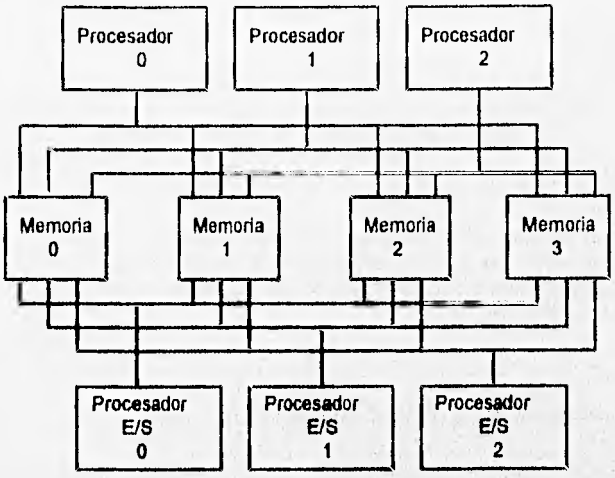


Figura 1.18 Almacenamiento de interconexión múltiple

I.5.2 NIVELES DE PLANIFICACION

Para ordenar dentro del procesador la jerarquía de los procesos son utilizados los *"niveles de planificación"*. Dichos niveles se clasifican fundamentalmente en tres:

- *Planificación de alto nivel.* En esta etapa se determina a qué trabajos se les permitirá utilizar más activamente los recursos del sistema.
- *Planificación de nivel intermedio.* En esta etapa se determina a qué proceso se le permite tener posibilidad de utilizar la UCP.
- *Planificación de bajo nivel.* En esta última etapa se determina el proceso que puede utilizar la UCP y se le asigna.

La planificación como una forma de organización debe cumplir con objetivos precisos, los cuales pueden mencionarse de manera sencilla como sigue:

- Maximizar la capacidad de ejecución.
- Maximizar el número de usuarios que tienen acceso y que requieren tiempos de respuesta aceptables.
- Minimizar la sobrecarga.
- Equilibrar el uso de recursos.
- Asegurar prioridades.

Para poder lograr los objetivos planteados anteriormente, deben seguirse una serie de indicaciones:

- Es necesario limitar un proceso a las operaciones de entrada/salida.
- Es necesario limitar un proceso a la UCP.
- Hay que verificar si el proceso que se ejecuta es tipo lote o interactivo, considerando que un proceso interactivo puede ser en muchas ocasiones la petición de algún usuario aficionado y que por ello no sea tan importante. Si se considera un proceso por lote, estamos hablando de que el usuario no se encuentra presente y ello puede provocar retrasos considerables en sus procesos.
- Se debe verificar que tan necesaria puede ser una respuesta rápida.
- Se debe verificar la prioridad del proceso que se va a ejecutar.
- Hay que verificar con oportunidad el tiempo real de ejecución que ha recibido un proceso.

*** Temporizador de Intervalos o Reloj de Interrupciones**

Cuando nos encontramos con un proceso en la UCP decimos que es un proceso en ejecución. Si tenemos entonces un proceso del sistema operativo, decimos que se ejecuta el sistema operativo y éste puede tomar decisiones que tengan influencia en la operación del sistema. Si queremos evitar que, debido a lo anterior, los usuarios manipulen el sistema, podemos hacer uso de mecanismos para que la UCP no sea accesible a ellos.

El sistema operativo cuenta con un reloj de interrupción o temporizador de intervalos cuya función es la de generar una interrupción en algún tiempo específico. Si el usuario se encuentra ejecutando un proceso y el reloj lo interrumpe, esta interrupción provoca que entre en funcionamiento el sistema operativo y es él quien decide lo que se obtendrá de la UCP.

El reloj de interrupciones ayuda a garantizar tiempos de respuesta razonables a usuarios interactivos.

Las prioridades que reciba un sistema pueden ser asignadas automáticamente por el sistema o también ser asignadas del exterior, de manera racional o arbitrariamente.

Las prioridades estáticas no cambian, tienen una sobrecarga relativamente baja. Las prioridades dinámicas responden a cambios. Su implementación es más complicada y generan sobrecarga la cual se justifica por el crecimiento de respuestas del sistema.

*** Planificación de procesos por el método del Primero en Entrar, Primero en Salir (FIFO)**

En este tipo de planificación los procesos se ejecutan de acuerdo a su tiempo de llegada a la cola de listos. Cuando el proceso llega a la UCP, se ejecuta hasta que termina. Se utiliza muy poco en los esquemas de muestreo de los sistemas actuales (Ver Fig. 1.19).

Figura 1.19 Planificación utilizando FIFO

* Planificación de Asignación en Rueda (RR)

En este tipo de planificación los procesos se ejecutan como en FIFO, pero la cantidad de tiempo que se les otorga de UCP es limitada. A esto se le llama *división de tiempo o cuanto* (Ver Fig. 1.20).

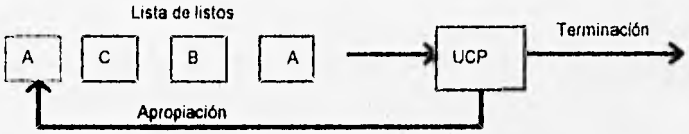


Figura 1.20 Planificación de asignación en rueda (RR)

*** Planificación del Trabajo más Corto Primero (SJF)**

Este tipo de planificación es no apropiativa. La planificación SJF favorece los trabajos más cortos, quitando oportunidad a los más largos; hace una selección de los trabajos y con ello se asegura de que el que se encuentre en ejecución termine lo más pronto posible. Sin embargo, SJF tiene el problema de no saber exactamente los tiempos que requiere cada proceso y lo que hace es confiar en los tiempos que el usuario estime para ello.

*** Planificación del Tiempo Restante más Corto (SRT)**

La planificación SRT es apropiativa, contraria a SJF, y es muy útil en tiempo compartido. El proceso espera de acuerdo al tiempo asignado de ejecución que sea más largo, siendo el siguiente en ser ejecutado, y ésta consideración incluye a los procesos que vayan llegando. Cuando el proceso comienza a ejecutarse no para hasta que termina.

SRT tiene una sobrecarga mayor que SJF, ya que requiere registrar el tiempo de servicio transcurrido.

*** Planificación del Siguiente con Relación de Respuesta Máxima (HRN)**

Este tipo de planificación corrige las deficiencias de SJF, especialmente la preferencia que se tiene a los trabajos cortos a costa de los largos.

De acuerdo a lo anterior, las prioridades dinámicas en HRN se calculan como:

$$P = \frac{te + ts}{ts}$$

donde: P = prioridad
te = tiempo de espera

ts = tiempo de servicio

I.6 ALMACENAMIENTO AUXILIAR O SECUNDARIO

I.6.1 OPERACION DE ALMACENAMIENTO DE CABEZA MOVIL

- Almacenamiento de disco de cabeza móvil

En la siguiente figura (Fig. 1.21) tenemos la representación esquemática, vista lateralmente, de un *disco de cabeza móvil.* Los datos se graban en los platos o discos magnéticos que se muestra, estos se conectan por medio de un eje común, girando a velocidades de 3600 rpm.

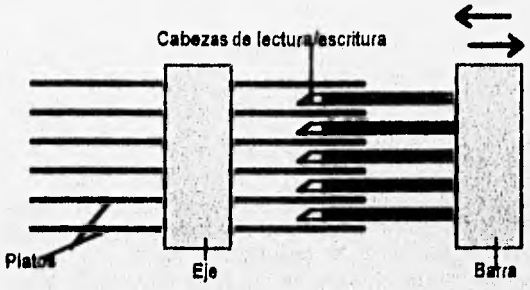


Figura 1.21 Esquema de un disco de cabeza móvil

La información puede ser escrita o leída por medio de una serie de cabezas de lectura/escritura, sólo una cabeza por cada superficie de disco. Los datos a los que puede acceder la cabeza son aquellos que tiene junto a sí, por lo que para poder ser leído algún dato, el disco debe de girar hasta encontrar la información adecuada, lo cual se realiza en un tiempo determinado (tiempo de latencia).

Todas las cabezas de lectura/escritura se encuentran montadas en una *barra* o *brazo móvil.* La barra tiene movimientos hacia afuera y hacia adentro, teniendo de esta forma, acceso a un mayor número de información. Para tener acceso a una posición en particular, la serie de pistas trazadas por todas las cabezas de lectura/escritura forman un cilindro vertical, por lo que al movimiento de la barra hacia un cilindro nuevo se le conoce como operación de búsqueda.

Al requerir posicionarnos en un registro específico, el sistema realiza varios movimientos:

1) La barra se mueve al cilindro adecuado.

2) La parte donde se encuentra guardado el registro debe de girar hasta encontrarse situado bajo la cabeza de lectura/escritura.

3) El registro (que puede variar en su tamaño), debe girar para que la cabeza realice la operación de lectura en este caso. El tiempo total de acceso a un registro específico, por lo regular es de 0.01 a 0.1 segundos.

I.6.2 CARACTERISTICAS DE LA PLANIFICACION

Debido a los procesos que se realizan al leer o escribir registros, obtenemos un tiempo de respuesta muy lento, ya que las peticiones de ejecución se llevan acabo mucho más rápido, en este caso las instrucciones se ejecutan una tras otra formando líneas de espera o colas.

Para reducir el tiempo utilizado en la búsqueda de registros se ordenan las colas de peticiones, a este proceso se le conoce como *planificación del disco.* Tal planificación implica realizar un examen a fondo de las peticiones pendientes para poder determinar la forma más eficiente de ejecutar las peticiones y que éstas puedan ser atendidas con el mínimo de movimientos mecánicos.

Existen dos tipos de planificación de uso común que son: *optimización de la búsqueda* y *optimización rotacional (o latencia).*

Algunas de las características que se desean llegar a obtener son:

- Capacidad de ejecución.
- Media del tiempo de respuesta.
- Varianza de los tiempos de respuesta (predecibilidad).

La planificación intenta elevar al máximo la capacidad de ejecución de peticiones por unidad de tiempo. Además, debe de minimizar el tiempo promedio de espera.

I.6.3 OPTIMIZACION DE LA BUSQUEDA

A continuación se presentan algunas estrategias de optimización de la búsqueda:

- *FCFS.* Se ejecutan los procesos en orden lineal, conforme se presentan se atienden; no existe reordenamiento de la cola de espera.

- *SSTF.* El brazo se posiciona en la siguiente petición que minimice el movimiento del brazo, realizando menor tiempo de búsqueda que la estrategia anterior.

- *SCAN.* El brazo se encuentra en movimiento de un lado a otro de la superficie del disco ejecutando todas las peticiones que encuentra. Sólo cambia de dirección cuando no encuentra más peticiones en la dirección actual.

- *C-SCAN.* El brazo del disco se mueve hacia adentro del disco en una sola dirección y va ejecutando las peticiones conforme las va encontrando, pero si no existen más peticiones el brazo salta a la más cercana de la parte externa del disco.

- *SCAN de n-pasos.* El movimiento del brazo se realiza igual que en SCAN, pero en este caso todas las peticiones que se obtengan en esa dirección se colocan en lotes y posteriormente se ordenan para realizar un mejor servicio durante el movimiento de retorno.

- *Esquema Eschenbach.* El movimiento del brazo se realiza igual que en C-SCAN, pero con excepción de que cada cilindro es atendido por una pista completa de información, existan o no peticiones para tal cilindro. Las peticiones se reordenan dentro de un cilindro obteniendo ventaja sobre la posición rotacional, pero si dos peticiones cambian posición de sectores dentro de un cilindro, sólo se atiende una en el movimiento actual del brazo.

I.6.4 OPTIMIZACION ROTACIONAL

En condiciones de alto número de peticiones es cuando se utiliza la optimización rotacional, puesto que cuando se tiene un gran número de peticiones, aumenta la probabilidad de que existan varias peticiones pendientes para el mismo cilindro. Por lo que se utiliza la estrategia SSTF, la cual analiza todas las peticiones y atiende primero a las que tienen retraso rotacional más pequeño.

I.6.5 SISTEMAS DE ARCHIVOS Y BASES DE DATOS

Un *archivo* es un grupo de datos etiquetados y pueden ser manejados como unidad de operaciones, realizando con ellos las siguientes funciones:

- *Open*: se prepara un archivo a ser llamado.
- *Close*: se evitan más llamadas a un archivo hasta abrirlo nuevamente.
- *Create*: crear un nuevo archivo.
- *Destroy*: destruir un archivo.
- *Copy*: crear una copia del archivo con diferente nombre o con el mismo.
- *Rename*: cambiar el nombre de un archivo.
- *List*: desplegar el contenido de un archivo.

Los elementos de datos individuales dentro de un archivo se pueden manejar con las siguientes operaciones:

- *Read*: Se introduce un dato de un archivo a un proceso.
- *Write*: Introduce un dato de un proceso en un archivo.
- *Update*: Modifica un dato existente.
- *Insert*: Añade un dato nuevo.
- *Delete*: Elimina un dato de un archivo.

*** Funciones del Sistema de Archivos**

Algunas funciones del sistema de archivos son las siguientes:

1) El usuario puede crear, modificar y eliminar archivos.

2) El usuario puede compartir sus archivos de forma controlada para

basarse en el trabajo de los demás.

3) El mecanismo encargado de compartir los archivos debe controlar el *acceso de lectura*, *de escritura*, *de ejecución* o combinaciones de estos.

4) El usuario puede estructurar sus archivos de la forma más adecuada para él.

5) El usuario puede ordenar la transferencia de información entre archivos.

6) Se deben de proporcionar las posibilidades de *respaldo* y *recuperación*, como prevención contra pérdida de datos.

7) El usuario puede hacer referencia de sus propios archivos por medio de *nombres simbólicos*.

8) El sistema de archivos debe de proporcionar posibilidades de *cifrado* y *descifrado*.

9) El sistema de archivos debe de proporcionar un acoplamiento favorable al usuario.

*** Descriptor de Archivos**

El descriptor de archivos es un bloque de control que contiene información que el sistema necesita para administrar el archivo. Los descriptores de archivos se encuentran en el almacenamiento secundario e incluyen:

- Nombre simbólico del archivo.
- Localización del archivo en almacenamiento secundario.
- Organización del archivo.
- Tipo de dispositivo.
- Datos de control de acceso.
- Tipo de archivo.
- Disposición.
- Fecha y tiempo de creación.
- Fecha de destrucción.
- Fecha de última modificación.
- Suma de las actividades de acceso.

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CAPITULO II : SOFTWARE Y HARDWARE

Dentro del área de la computación encontramos dos grandes ramas: Hardware y Software, que a su vez contienen una gran cantidad de elementos que las forman como parte importante de la computación.

Muchos sistemas operativos fueron desarrollados durante la década de los 60's y eran escritos por personas que no entendían realmente que su software y hardware debía recibir el aporte de la ingeniería para ser confiables, comprensibles y tener facilidad de mantenimiento. Los errores causados en la primeras etapas de los proyectos no se localizaban sino hasta mucho después de que los sistemas habían sido entregados a los clientes. Se dedicó tanta atención a estos problemas, que cientos de científicos computacionales e industriales comenzaron a dedicar considerables recursos a los problemas de la construcción de sistemas de software. Esto acrecentó el desarrollo del campo de la ingeniería de software.

El surgimiento de la ingeniería de software y el reconocimiento de la importancia de lograr un acercamiento disciplinado y estructurado a la construcción del mismo, que fuese confiable, entendible y con facilidad de mantenimiento, han sido fomentados por las experiencias de muchos de los esfuerzos de desarrollo de sistemas operativos de esta década.

II.1 SOFTWARE

En computación el software es la parte que se refiere a los programas, instrucciones, paquetes y lenguajes de programación que son hechos a base de algoritmos para la resolución de problemas y/o la simplificación de trabajos. El software consta de diferentes elementos:

- El lenguaje de máquina es aquél que está diseñado para que la computadora comprenda directamente las instrucciones que recibe. Generalmente, este lenguaje es un cierto código de caracteres diseñado para el equipo que se maneje.
- Los lenguajes ensambladores surgen de la necesidad de aumentar la velocidad de programación y reducir los errores de codificación. Son programas sencillos que evitan la escritura directa de un lenguaje de máquina, pero deben ser traducidos a éste para que las instrucciones puedan ser procesadas debidamente.
- Los compiladores surgen de la necesidad de perfeccionar los ensambladores, así los lenguajes de alto nivel permiten a los usuarios escribir programas en un modo independiente del lenguaje de máquina, esto es, que los usuarios pueden dedicarse solamente a la resolución de sus problemas sin tener que preocuparse por escribir además un código para que la máquina "comprenda" lo que se le pide. Los lenguajes de alto nivel son traducidos al lenguaje de máquina a través de los compiladores.
- Existe también un sistema desarrollado para reducir las complejidades en el control de las entradas/salidas de diversas rutinas de operación de canales, y éste es llamado "*sistema de control de entrada/salida* (IOCS)".

Se tienen dos aplicaciones fundamentales para los lenguajes de alto nivel: la primera se refiere a los lenguajes orientados al procedimiento (PASCAL, COBOL, FORTRAN, BASIC); y

la segunda a los lenguajes que están orientados al problema (GPSS, para simulación; SPSS, para cálculos estadísticos).

Los cargadores son programas que colocan las instrucciones y datos de un programa en el almacenamiento principal y para que los programas sean ejecutados deben de ser colocados en el mismo. Un cargador absoluto pone estos artículos en las localidades específicas indicadas en el lenguaje de máquina. Un cargador de relocalización puede cargar un programa en varios lugares del almacenamiento principal. Un cargador de enlace, al igual que un editor de enlace, realiza el proceso de combinación de programas. El cargador de enlace combina cualquiera de los programas requeridos y los carga directamente al almacenamiento primario. El editor de enlace hace la misma función, sólo que simula que el almacenamiento secundario sea la referencia a seguir posteriormente.

II.2 HARDWARE

El hardware es todo lo que se refiere a la parte física de la computadora; como pueden ser dispositivos de entrada/salida, conexiones, procesadores, etc.

Dentro del hardware encontramos diferentes elementos con funciones específicas, como se explica a continuación:

- La compaginación del almacenamiento se refiere a la colocación de localidades de memoria adyacentes en diferentes bancos de almacenamiento, lo que permite tener la opción de varias referencias al mismo tiempo. Las referencias secuenciales simples al almacenamiento principal se hacen por orden, lo que permite tener acceso a varios datos al mismo tiempo siempre que estos se encuentren en diferentes bancos de memoria.
- Un registro de relocalización permite localizar nuevamente programas de una forma dinámica. La dirección base de un programa en la memoria principal se encuentra en el registro de relocalización.
- El escrutinio es el procedimiento que permite que una unidad de disco verifique el estado de otra de funcionamiento independiente. El método es sencillo: la primera unidad verifica que la segunda se encuentre en estado conveniente, si no se cumple esto, la primera prosigue con lo que estaba haciendo. Las interrupciones permiten que una unidad dé atención inmediata a otra, de manera que la primera cambie de estado.
- El buffer es un área de almacenamiento de datos. Mientras se lleva a cabo dicho almacenamiento no puede darse ningún procesamiento de los mismos, de igual manera, mientras se procesan los datos no puede llevarse a cabo recepción alguna de datos adicionales.
- Los dispositivos periféricos permiten el almacenamiento de gran cantidad de información fuera del almacenamiento primario (memoria interna) de la computadora. Entre los dispositivos periféricos más conocidos encontramos los discos magnéticos.
- Los temporizadores son elementos que permiten el acceso a varios usuarios en un procesador en intervalos de tiempo.
- Los canales de entrada/salida son elementos muy útiles en el manejo de la computadora, gracias a ellos se puede tener acceso directo al almacenamiento principal para almacenar o recuperar información sin necesidad de esperar un determinado tiempo para cada proceso. En los sistemas actuales manejados por interrupciones, los canales de entrada/salida son habilitados con mayor facilidad para su funcionamiento.

- Los sistemas de almacenamiento virtual permiten a los programas referenciar direcciones que no necesariamente deben corresponder con las direcciones reales disponibles en el almacenamiento primario.

- En el multiprocesamiento varios procesadores comparten un almacenamiento primario común y un sólo sistema operativo.

- El acceso directo a memoria es una manera de reducir al mínimo las interrupciones durante la ejecución de programas ya que sólo requiere de una interrupción por cada bloque de caracteres transferidos durante la entrada/salida.

- Se le llama canalización a una técnica de hardware que permite que varias instrucciones puedan estar al mismo tiempo en un estado de ejecución.

*** Separación de Costos del Software y el Hardware**

En los primeros años de la década de los 70's, los distribuidores de computadoras solían vender un solo tipo de producto: el hardware. Los sistemas operativos, programas de soporte, sistemas de aplicaciones y los manuales de documentación y educación se entregaban a los usuarios sin cargo alguno. Como consecuencia, los proveedores tendían a ver el software como un artículo de "regalo" y se negaban a responsabilizarse de su calidad.

Los vendedores publicaban largas listas de errores conocidos como una "cortesía" hacia sus clientes. Aquellos clientes que llamaban a los vendedores quejándose acerca de algún problema, generalmente eran reprendidos indicándoles que de haber leído la lista de errores conocidos habrían evitado en primer lugar utilizar esa función.

Por otra parte, IBM desempaquetó costos del software y hardware, esto significaba que ambos se cobrarían por separado, aunque continuó suministrando algún software básico sin cargo. Los efectos sobre la industria fueron significativos: ahora era mucho más difícil para los distribuidores de computadoras insertar las largas cláusulas en sus contratos que los excluía de responsabilidades. Casi de la noche a la mañana, como una respuesta a esta separación de costos, se creó una industria independiente de software. Otros fabricantes siguieron la tendencia y la separación de costos no tardó en ampliarse a toda la industria. Los usuarios recibieron gran incentivo para buscar y comprar su software entre los muchos productos competitivos disponibles.

Una consecuencia importante de la separación de costos fue que los distribuidores comenzaron a diseñar su software más modular, de manera que pudieran venderse como unidades individuales.

El mercado de computadoras compatibles con IBM tuvo gran auge. El hardware de computadoras continuó declinando en precio, mientras que las velocidades de procesamiento y capacidades de almacenamiento aumentaban, y el tamaño físico de los procesadores y memorias disminuía.

La "escala de integración" (disminución de tamaño de circuitos) continuaría aumentando con el VLSI (*Integración a escala muy grande*) moviéndose hacia un ULSI (*Integración a escala ultragrande*) en la siguiente década. El multiprocesamiento se volvería más común. Muchas de las funciones de los sistemas operativos realizadas entonces por software tendería hacia el microcódigo. Las arquitecturas del hardware del futuro distribuiría el control entre procesadores localizados.

Los lenguajes estaban siendo desarrollados con la finalidad de explotar la concurrencia, y el hardware y los sistemas operativos están siendo diseñados para ejecutar con mayor eficiencia los programas concurrentes que requieren de un procesamiento paralelo.

El paralelismo masivo es más común, siendo posible ejecutar programas paralelos a mucha mayor velocidad debido al alto grado de concurrencia.

Los computadoras y sus sistemas operativos eran diseñados para fomentar la operación de máquinas virtuales. Las máquinas reales se encuentrán ocultas para los usuarios.

El concepto de familias de computadoras continua tal y como fue introducido por las series 360 de IBM.

A medida que los fabricantes introducían nuevas generaciones de computadoras, los programas existentes se ejecutaban en los nuevos equipos tal y como estaban o con pequeños cambios.

Los desarrollos en la ingeniería de software dieron por resultado sistemas operativos más preservables, confiables y comprensibles.

El costo de la comunicación de datos continuaba disminuyendo y las velocidades de transmisión de datos aumentaban.

Las computadoras estaban conectadas cada vez más a redes de sistemas, y el trabajo del usuario podía ser realizado por una computadora de la cual éste no tenga conocimiento. El concepto de almacenamiento virtual se utilizará por mucho tiempo más. La perspectiva previa de un sistema operativo como administrador de recursos perdurará.

El concepto de proceso distribuido provocó que fueran desarrollados sistemas operativos dispersos, en lo cuales las funciones de los sistemas operativos son distribuidas entre varios procesadores a través de grandes redes de sistemas.

II.3 MEMORIA FIJA

Dentro del desarrollo de la computación nos encontramos con diversos avances significativos en el área. Así, nos introducimos al concepto de "*microprogramación*" el cuál se refiere a elaborar programas que de alguna manera queden almacenados en la computadora. A finales de la década de los 70's, se creó la microprogramación dinámica lo que permitió cargar microprogramas dentro del almacenamiento de control desde el cual son ejecutados.

Un *microcódigo* es un programa elaborado para mejorar el rendimiento de ejecución de los sistemas computacionales y quedan contenidos en la memoria fija de la computadora.

La *emulación* es una técnica por medio de la cual se hace que una máquina aparente ser otra. Las instrucciones del lenguaje de máquina que va a ser emulada se microprograman en lo que se llama "*maquina anfitriona*". Una vez que esto se realiza, el lenguaje de máquina de la máquina emulada puede ser leído por la anfitriona.

Los microprogramas tienen más acceso al hardware que los programas de lenguaje de máquina por ello es más fácil detectar y corregir errores; sin embargo, algunos sistemas combinan el microdiagnóstico con las instrucciones del lenguaje de máquina, con lo cual se logra una revisión más confiable.

También se han desarrollado diversos tipos de computadoras atendiendo los problemas personales de los usuarios, de esta forma nos encontramos con el funcionamiento de computadoras personales que cuentan con los elementos necesarios para la solución de problemas concretos que se puedan plantear. Sin embargo, una mejor adaptación de los usuarios a los sistemas computacionales se ha logrado a través del software. El software proporciona una serie de programas a ejecutar por medio del hardware, los programas de éste adaptan las computadoras de acuerdo a las necesidades del usuario.

El acceso a los microcódigos permite una serie de rutinas de interrupción con el fin de lograr mejores resultados en la ejecución de programas. La implementación de funciones del sistema operativo en microcódigo puede mejorar el rendimiento del sistema computacional, reducir costos en el desarrollo de programas y aumentar la seguridad del sistema.

La microprogramación es un campo amplio de estudio y se refiere a la realización de programas residentes en la memoria fija de la computadora, que tienen como prioridad permitir una mejor comunicación usuario/máquina utilizando un mínimo de espacio de almacenamiento. La microprogramación consiste en una serie de instrucciones específicas que la máquina tiene definidas.

II.4 DIRECCIONAMIENTO

* Estructura General

Casi todas las computadoras convencionales modernas se basan en el concepto de "computadora de programa almacenado" o "máquina de direcciones", atribuido generalmente al matemático John Von Newman (1903-1957); esta estructura se muestra en la figura 2.1

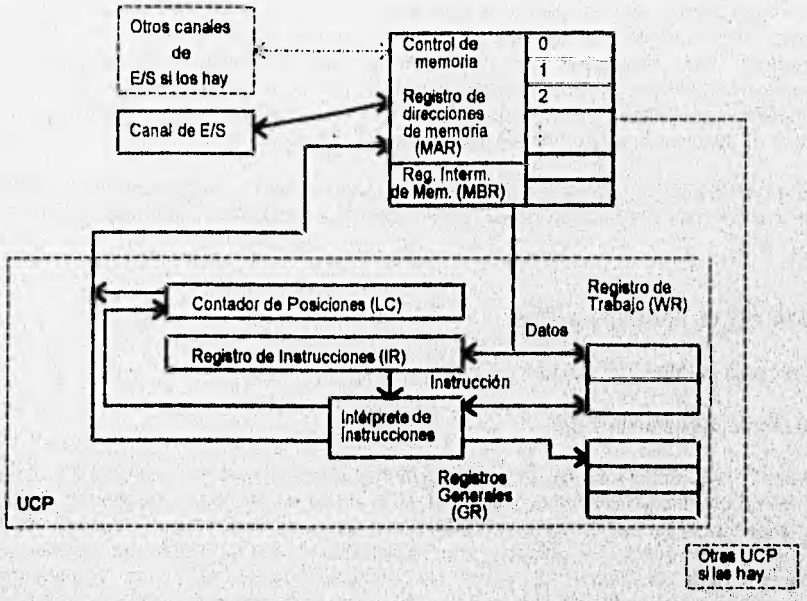


Figura 2.1 Estructura General de una Máquina

La UCP consiste en un intérprete de instrucciones, un contador de posiciones, un registro para instrucciones, varios registros de trabajo y registros generales. El intérprete de instrucciones es un grupo de circuitos electrónicos, que realiza la interpretación de las instrucciones que obtiene de la memoria. El contador de posiciones (LC, Location Counter), también llamado *"contador de programa"* (PC, Program Counter) o *"contador de instrucciones"* (IC, Instruction Counter), es un dispositivo de memoria física que almacena la posición de la instrucción que se está realizando, almacenándose una copia de ésta en el registro de instrucciones (IR, Instruction Register). Los registros de trabajo son dispositivos de memoria que se utilizan como "borrador" por parte del intérprete, mientras que los registros generales son utilizados por el programador como posiciones de almacenamiento para funciones especiales.

La interfaz primaria entre la memoria y la UCP se realiza por medio del registro de direcciones de memoria (MAR, Memory Address Register) y del registro intermediario de la misma. El MAR contiene la dirección de la posición de memoria que debe leerse o cargarse. El registro intermedio de memoria (MBR, Memory Buffer Register) contiene una copia de la posición de memoria especificada por el MAR después de una lectura, o el nuevo contenido de la posición de memoria después de una escritura. El control de memoria (memory controller) es un dispositivo físico que transfiere datos entre el MBR y la posición de memoria cuya dirección esta almacenada en el MAR.

La unidad básica de memoria es el byte esto es, ocho bits de información. En otros términos, cada posición de memoria direccionable puede contener hasta ocho bits de información. Las direcciones de la memoria admiten hasta tres componentes. Específicamente, el valor de una dirección es igual al valor de un desplazamiento (offset) más el contenido de un registro base, más el contenido de un registro índice. En general, las unidades de memoria se indican dando la dirección de su byte de menor orden.

Los registros de uso general pueden ser empleados para diversas operaciones aritméticas y lógicas, y como registros bases. Como registros bases sirven para formar direcciones.

II.5 FAMILIAS DE PROCESADORES

* Evolución del Microprocesador

- *El Microprocesador de 4 bits.*

En 1971, Intel Corporation dió a conocer al mundo el primer microprocesador de 4 bits: el 4004. Es un controlador programable sobre un chip, el cual es muy pobre comparado con los estándares de hoy en día. El conjunto de instrucciones contenidas en este microprocesador son sólo 45 diferentes. Lo anterior da como resultado que el 4004 sea utilizado en aplicaciones muy limitadas, tal como los primeros juegos de video y pequeños microprocesadores basados en controladores. Si se requieren aplicaciones más sofisticadas, el 4004 se vuelve inadecuado.

- El microprocesador de 8 bits.

Más tarde, Intel se dió cuenta que el microprocesador podría ser un producto comercial, por lo que liberó el 8008 de 8 bits (el cual fué antecedente del 8080A). A este nuevo microprocesador se le expandió la memoria (16K x 8), teniendo un total de 48 instrucciones disponibles, dando la oportunidad de tener aplicaciones mucho más avanzadas.

Los ingenieros comenzaron a desarrollar más y más, demandando un mayor uso de los microprocesadores; sin embargo, la aún pequeña memoria y el conjunto de instrucciones del 8008 pronto comenzó a tener sus límites de uso. Así, en 1973 Intel introdujo al mercado el 8080, siendo éste el primer microprocesador moderno de 8 bits.

El 8080 tiene más direcciones de memoria y ejecuta más instrucciones que el 8008, pero la gran diferencia entre ambos es que el primero ejecuta las instrucciones diez veces más rápido (2 μs) que el segundo (20 μs). El 8080 utiliza tecnología compatible con TTL (Transistor-Transistor Logic). Todas estas ventajas se introdujeron en la era del 8080 y del microprocesador.

Una nueva versión del 8080 es el 8085, que fue liberado por Intel en 1977. Sólo un poco más avanzado que el 8080, el 8085 direcciona la misma cantidad de memoria, ejecuta el mismo número de instrucciones y le toma 1.3 μs en vez de 2.0 μs. La principal ventaja del 8085 es que éste tiene incorporado un generador de reloj y un sistema controlador que posee componentes externos basados en el sistema del 8080.

- El microprocesador de 16 bits.

En 1978, Intel liberó el microprocesador 8086 y un año después el 8088. Ambos son microprocesadores de 16 bits que ejecutan instrucciones en tiempos de 400 ns, mejorando la velocidad de ejecución del 8085. Además, el 8086 y el 8088 son capaces de direccionar 1 Mbyte o 512 K palabras (16 bits) de memoria. Estas velocidades de ejecución y tamaños de memoria permiten que el 8086 y el 8088 reemplacen pequeñas minicomputadoras en muchas aplicaciones que se encuentran limitadas en memoria y tamaño de palabra.

Una de las necesidades importantes en la evolución de los microprocesadores de 16 bits ha sido la realización de multiplicaciones y divisiones, estas funciones no se encuentran disponibles en la mayor parte de los microprocesadores de 8 bits, con excepción del MC6809 de Motorola que puede multiplicar pero no dividir.

Para 1981, cuando IBM seleccionó el 8088 para incorporarlo a su computadora personal, la arquitectura 8086/8088 tomó la dirección definitiva para convertirse en el estándar de la industria de la computadora personal.

*** Arquitectura básica del 8086 y 8088**

En la figura 2.2 se ilustra la operación normal de un 8085, el cual es un microprocesador típico de 8 bits. Se debe de observar que las instrucciones son llamadas de la memoria por medio de una operación de lectura de memoria. Después, mientras que el 8085 ejecuta la instrucción, el sistema de memoria es desocupado. El 8086 y 8088 hacen uso del tiempo de memoria desocupada o libre para ejecutar la siguiente instrucción mientras se ejecuta la actual.

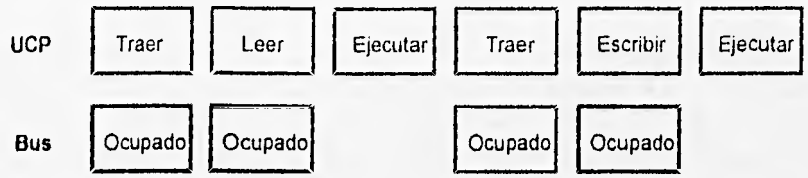


Figura 2.2 UCP 8085A Indicando el Nivel de Actividad del Bus

En la figura 2.3 se muestra la secuencia de ejecución de los microprocesadores 8086 y 8088. Puede apreciarse que el bus está siempre ocupado. Además, se debe observar que ambos microprocesadores contienen dos unidades internas, la *unidad de ejecución* (EU, execution unit) y la *unidad de interfaz de bus* (BIU, bus interface unit). El BIU es el responsable de ir por una instrucción (el operando de una instrucción o dato de memoria), y la EU es la responsable de ejecutar las instrucciones. El 8086 y el 8088 son capaces de utilizar el bus con un máximo de eficiencia porque ambos contienen memoria interna en la forma de una *cola* o memoria FIFO.

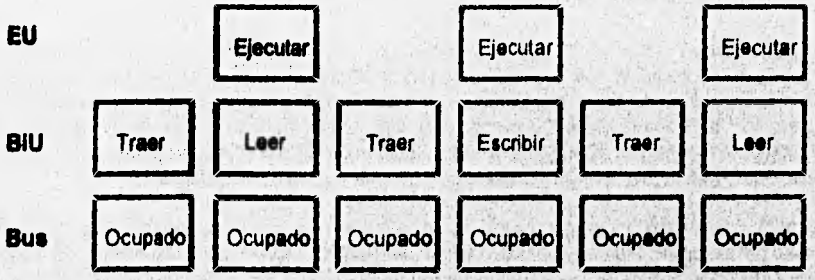


Figura 2.3 Los Microprocesadores 8086 y 8088 Indicando el Nivel de Actividad de su Bus

La figura 2.4 muestra la localización de la EU y de la BIU. La cola del 8086 es de 2 bytes de ancho y tres localidades de profundidad. Así es capaz de manipular 3 números de 16 bits. El 8088 tiene una cola de un byte de ancho y 4 bytes de profundidad. Las colas permiten que el 8086 y el 8088 traigan instrucciones, mientras que la EU está ocupada ejecutándolas. Esto permite que los microprocesadores 8086/8088 utilicen el sistema de memoria más eficientemente.

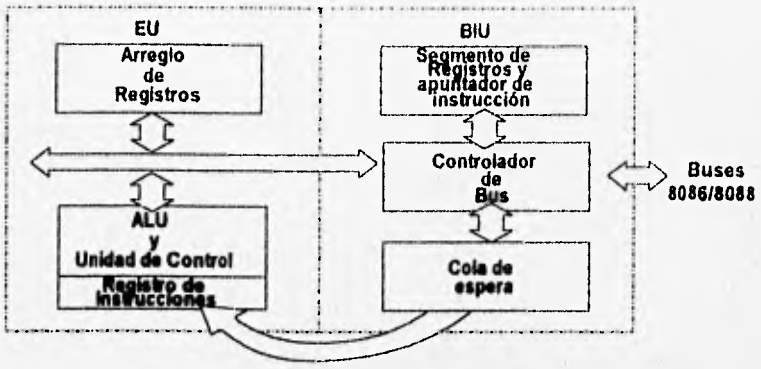


Figura 2.4 Estructura Interna de los Microprocesadores 8086 y 8088

El 8088 contiene internamente cerca de 29000 transistores y opera a 4.77 MHz; las operaciones que realizan se llevan a cabo por movimiento de bloques de datos en espacios precisos de tiempo. Las PC-XT pueden tener un ciclo de bloques de 8 bits, utilizándose para formar un caracter simple del alfabeto; esto puede resultar muy lento en aplicaciones gráficas. Las XT turbo pueden soportar operaciones más rápidas de lo normal (4.77 MHz); pueden operar en 8, 10 y 12 MHz.

*** Arquitectura del Sistema**

En la figura 2.5 se aprecian los diagramas de pines de los microprocesadores 8086 y 8088. En ambos casos la comunicación del sistema ocurre a través de tres buses: *dirección, datos* y *control*. El bus proporciona una dirección de memoria para el sistema de la misma y también las direcciones de entrada/salida para el sistema de entrada/salida. El bus de datos transfiere a estos entre el microprocesador y la memoria, juntando el sistema de entrada/salida. El bus de control proporciona señales de control que causan que la memoria o la entrada/salida ejecuten alguna operación de lectura (*RD*) o escritura (*WR*).

El ancho del bus de datos de ambos microprocesadores es diferente ya que en el 8086 se utiliza un bus de 16 bits, mientras que en el 8088 se emplea un bus de 8 bits. Las direcciones del bus del 8086 contienen una señal BHE (*bus high enable, habilitación alta del bus*), y el bus de direcciones del 8088 no tiene. BHE es utilizado para seleccionar la parte superior de los bits de datos siempre que ocurra la lectura o escritura de un byte, y la dirección de bit (A_0) es usada para seleccionar el byte más bajo para una lectura de byte.

Otra diferencia entre estos dos microprocesadores son las señales *IO/M* en el 8088 (pin 28) y la *M/IO* en el 8086 (pin 28). La señal *IO/M* es usada para elegir el sistema de entrada/salida o la memoria. Si es un cero lógico se selecciona la memoria, si es un 1 lógico se elige el sistema de entrada/salida. La función del pin *M/IO* es idéntica con excepción de los niveles lógicos, los cuales se encuentran invertidos.

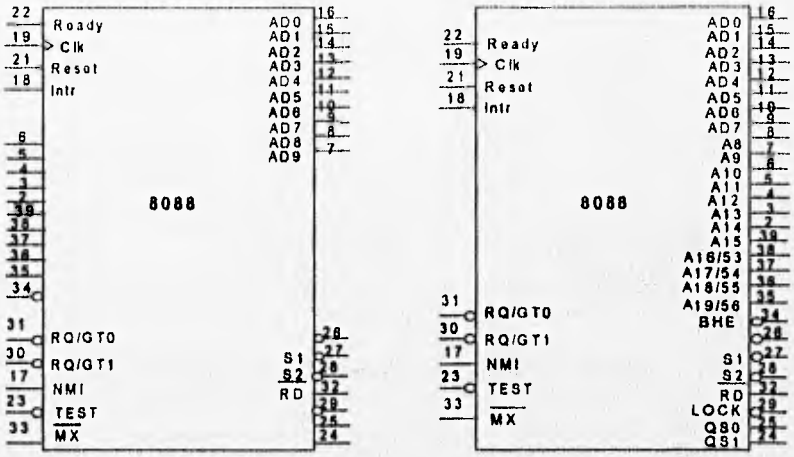


Figura 2.5 Diagrama de Pines de los Microprocesadores 8086 y 8088

El bus de datos de 16 bits del 8086 permite ir por instrucciones más rápidamente que el 8088. Ambos microprocesadores direccionan 1 Mbyte de memoria ya que los dos contienen un bus de direcciones de 20 bits.

• El Coprocesador Matemático 8087

La familia de coprocesadores 8087 ejecuta operaciones aritméticas y comparaciones sobre una variedad de tipos de datos. También es capaz de realizar funciones trascendentales como tangentes y logaritmos.

La familia 8087 puede multiplicar, sumar, restar, dividir, obtener raíz cuadrada, tangente parcial, arco tangente parcial y logaritmos. Los tipos de datos operados sobre el 8087 incluyen números enteros de 16, 32 y 64 bits; 10 dígitos de BCD (Binary Coded Decimal, sistema decimal codificado en binario) y números de punto flotante de 32, 64 y 80 bits. Las operaciones realizadas por el 8087 generalmente se ejecutan cerca de 100 veces más rápido que las operaciones equivalentes escritas por el más eficiente 8086/8088.

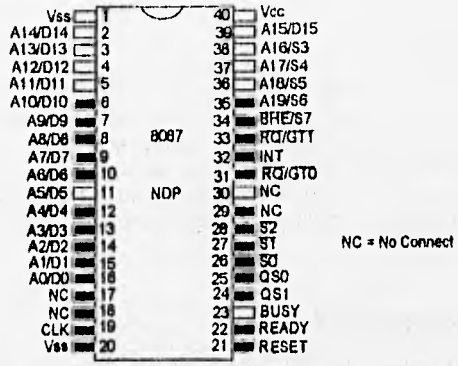


Figura 2.6 Diagrama de Pines del Coprocesador Matemático 8087

El 8087 es un coprocesador diseñado para correr en paralelo excediendo la rapidez del software escrito para los microprocesadores 8086 y 8088. El 8087 ejecuta 68 instrucciones numéricas con los microprocesadores 8086/8088. Estos últimos ejecutan todas las instrucciones normales, mientras que el 8087 ejecuta las suyas. Otros miembros de la familia incluyen a los 80187, 80287 y 80387. Las interfaces 80187 son para el 80186/80188, el 80287 para el 80286 y el 80387 para el 80386. En suma, al 80387 (una versión de 16 bits), el 80387SX es la interfaz para el 80386SX. El 80486 incluye un 80387 construido dentro del mismo circuito integrado.

*** Los Microprocesadores 80186, 80188 y 80286**

En el diseño de la PC-XT se realizó un nuevo diseño lógico por medio de la arquitectura del 8086. En 1982, Intel desarrolló dos nuevas implementaciones para cubrir las necesidades de mercado: los microprocesadores 80186 y 80286.

El chip altamente integrado del 80186 se desarrolló para el mercado del llamado *Inserto de Control* (Embedded Control) como aplicación para autos y controles de propósito especial.

Los microprocesadores Intel 80186, 80188 y 80286 son versiones mejoradas de los 8086/8088. El 80186 y el 80188 como los 8086/8088 son casi idénticos, la única diferencia entre el 80186 y el 80188 es la anchura de sus buses de datos. El 80186 (como el 8086) tiene un bus de datos de 16 bits y el 80188 (como el 8088) tiene un bus de datos de 8 bits. El registro interno de la estructura de los 80186/80188 y los 8086/8088 son virtualmente idénticas, la única diferencia es que los primeros tienen algunas reservas adicionales de

vectores de interrupción que no son usadas en los microprocesadores 8086/8088, y algunas poderosas incorporaciones de entrada/salida.

Como el 8086, el 80186 contiene una BIU y una EU, pero más grandes. Para la BIU y la EU, el 80186 tiene un generador de reloj, una interrupción de controlador programable, un programador de tiempo, un controlador programable para DMA (*Direct Memory Access*) y un chip programable para selección de unidad. Estos mejoramientos son considerablemente incrementados en la ejecución del 80188 y reducen el número de elementos requeridos de componentes periféricos necesarios para implementar un sistema.

El microprocesador 80288 es una versión avanzada del microprocesador 8086 que fue diseñado para multiusuarios y ambientes de multitareas. El 80288 puede direccionar 16 Mbytes de memoria física y 1 Gbyte de memoria virtual utilizando la unidad de dirección de memoria que está localizada dentro del microprocesador, el cual encuentra un extenso uso en las máquinas clones de tipo AT, que han inundado el mercado de las computadoras.

*** Los Microprocesadores 80386 y 80486**

En 1985 se presentó al mercado el microprocesador 80386, que es una versión completa de 32 bits de los microprocesadores 8086/8088, exáctamente como el 80286 es una versión de 16 bits.

El 80386 se caracteriza por la realización de multitareas, administración de memoria, memoria virtual con o sin paginación, protección de software y un sistema de memoria grande. Todo el software escrito para los 8086/8088, 80186/80188 y 80286 funcionan en el 80386, haciendo que este microprocesador sea compatible con las tempranas versiones de microprocesadores. En el caso del 80386, la cantidad de memoria que direcciona ha sido incrementada de 1 Mbyte en los 8086/8088 y 80186/80188, y 16 Mbytes en el 80286, a la enorme cantidad de 4 Gbytes. Estos 4 Gbytes (1 Gbyte = 1024 Mbytes) tienen un alcance de memoria física que puede direccionarse arriba de los 64 Tbytes (1 Tbyte = 1024 Gbytes) de memoria virtual disponible. Una memoria de 4 Gbyte puede almacenar aproximadamente 1 000 000 de páginas mecanografiadas (4 Kbytes por página) de información.

Existen dos tipos de 80386 disponibles: el 80386DX cuya versión es la más completa de este poderoso microprocesador. Una versión de bus de datos reducida (bus de datos de 16 bits) se encuentra disponible en el 80386SX. La principal diferencia entre el DX y el SX es que el segundo tiene un bus de datos de 16 bits y está diseñado para ser un reemplazo del microprocesador 80286. El bus de datos de 16 bits se diseñó para incorporar al 80386SX dentro de la tarjeta, que aloja a un 80286 con un mínimo de cambios para el circuito principal de la tarjeta.

Durante 1989 Intel introdujo el microprocesador 80486, el cual es básicamente un 80386 que contiene un coprocesador matemático 80387 y una memoria caché interna de 8Kbyte. Este alto nivel de integración causó que el 80486 tomara la delantera con respecto a los microprocesadores del tipo 8086.

El 80486 ejecuta más instrucciones que el 80386 en tiempos. El 80486 es un microprocesador que puede contener arriba de un millón de transistores; dentro de este circuito integrado se encuentra la unidad de administración de memoria MMU (*memory-management unit*), que es un coprocesador matemático compatible con las series 80X87; posee alta rapidez en memoria caché con capacidad de 8 Kbytes y un microprocesador de 32 bits que es compatible con los primeros.

El 80486 es capaz de ejecutar el software escrito para el microprocesador 8088 de la misma manera que los otros miembros de la familia, incluyendo los coprocesadores matemáticos 80X87 sin ninguna modificación. Por supuesto que este microprocesador contiene algunas instrucciones no presentes en los anteriores.

La RISC (*reduced instruction set computer*) o computadora de conjunto de instrucciones reducidas, diseñada para la 80486, reduce la cantidad de tiempos requeridos para ejecutar muchas instrucciones en sólo un ciclo de reloj. Esto proporciona un mejoramiento importante en la rapidez de ejecución de muchos programas, además de que alcanza un alto desempeño sin comprometer la compatibilidad del software.

En 1990 salió a la venta el primer chip de alta integración con la arquitectura 386: el microprocesador PC-portátil 80386SL. Cuando dicha arquitectura se combina con el 80360SL, se implementa el sistema ISA (Industry Standard Architecture bus) en un solo chip, resultando una PC portátil que mantiene 100% de compatibilidad con el software de sus antecesores.

En 1991 se dió a conocer el microprocesador 80486SX que permite un costo accesible para terminales de entrada del tipo negocios, con la tecnología 486. También en el mismo año Intel introdujo el microprocesador de 50 MHz 486DX, que mejoró la cobertura de la familia 486DX en un 50%.

El microprocesador Intel 486DX2 desarrollado en 1992 se creó para obtener un alto desempeño en las PC's, utilizando una técnica de *doblaje de velocidad* para facilitar el diseño.

La UCP llamada 486SX opera en el rango de 16, 20, 25 y 33 MHz mientras que la 486DX opera en 25, 33 y 50 MHz. El microprocesador de alta velocidad 486DX2 opera en 50 y 66 MHz. Muchas funciones a nivel de sistema están integradas en este último chip, incluyendo un procesador de 32 bits con un excelente conjunto de instrucciones y una diversidad de modos de direccionamiento que lo hacen flexible.

*** Microprocesador Pentium**

El microprocesador Pentium es un procesador considerado de quinta generación. Se comenzó a utilizar en marzo de 1993, realizandose una producción a gran escala del mismo. El Pentium es cinco veces más poderoso que su antecesor 486DX de 33 Mhz, y mantiene compatibilidad con el banco de software instalado más grande del mundo, es decir, con la familia X86, proporcionando a los usuarios una potencia computacional aumentada para las aplicaciones de alto nivel, como son los servidores de alto volumen, análisis financiero complejo y los programas de diseño e ingeniería auxiliados por una computadora, así como una variedad de aplicaciones en las PC's, algunos ejemplos son el video de movimiento completo, reconocimiento de voz y procesamiento de imágenes.

Los sistemas operativos con las nuevas interfaces gráficas para usuarios (GUi), tales como Windows, OS/2, Windows NT, NEXTSTEP 486, UNIX y Solaris se verán beneficiados con el incremento en la velocidad de procesamiento. El Pentium ofrecerá también ventajas en diferentes áreas tales como la investigación científica, diseño e ingeniería (CAD/CAE) y análisis financieros, además de aplicaciones cliente/servidor que requieren una gran velocidad y eficiencia en el procesamiento de datos.

El procesador Pentium emplea la tecnología e innovaciones de ingeniería más avanzadas. Este procesador está fabricado utilizando un proceso BiCMOS de 0.8 micrones y utiliza la arquitectura superescalar RISC, además tiene dos unidades de ejecución de cinco fases y puede procesar hasta dos instrucciones en un ciclo de reloj. Las UCP Intel 486 y 386 tienen una unidad de ejecución; la UCP Intel 486 está diseñado con un RISC *integer core*, que ejecuta la mayoría de las instrucciones en un ciclo de reloj.

El Pentium presenta como principales características, dos unidades de caché de 256 K, que han mejorado considerablemente la capacidad de las operaciones de punto flotante; un bus externo de 64 bits, y está constituido por 3.1 millones de transistores, casi tres veces más que el 486.

La *unidad totalmente compatible de punto flotante* (UPF) incorpora algoritmos optimizados y un hardware de multiplicación, división y adición, con una ejecución simultánea de ocho etapas para realizar una operación de punto flotante por cada ciclo de reloj. La UPF tiene capacidad para ejecutar varias aplicaciones cinco o diez veces más rápido que las ejecutadas en un procesador 486DX de 33 MHz. Otras técnicas avanzadas de diseño, como la *branch prediction*, bus de gran capacidad de 880 bits para datos internos y los cachés *write-back*, sirven para mejorar la capacidad de cualquier software. El Pentium mantiene la compatibilidad con las generaciones anteriores.

Además, este procesador incluye un controlador de caché avanzado 82496, un caché 82491, el controlador de interrupciones 82489DX y el chip set PCIset 82430. El 82489DX proporciona apoyo al sistema de multiproceso, siendo ésta la primera implementación de la *arquitectura de controlador de interrupciones avanzado programable* (APIC). El PCIset Intel 82430 proporciona *capacidad de bus local* (PCL) a los sistemas desktop del procesador Pentium. Este incluye un procesador de caché/DRAM, un acelerador del bus local y una lógica del sistema con un puente de expansión de bus EISA o ISA, lo cual permitirá ofrecer diferentes clases de sistemas en un rango variado de precio/desempeño.

La tecnología de Video de Intel conocida como *Indeo Video*, es parte del software que permite la ejecución de secuencias de video (playback) en una PC, al posibilitar que cualquier computadora basada en el 486, pueda reproducir los archivos Indeo con solo una tarjeta adicional. El Pentium ofrece alta calidad en las imágenes de video con movimiento completo a 30 fps. La tecnología Indeo adapta en forma ideal la calidad, velocidad y resolución de la imagen, para aprovechar el hardware más poderoso sin ajustar los archivos actuales; facilita la creación de archivos de video debido a que soporta la captura de un solo paso; utiliza una tarjeta que contiene el procesador de video Intel i750, el cual consiste en dos componentes VLSI (Very Large Scale Integration) que implementan subsistemas de alto desempeño. El i750 Video Processor suplementa los procesos de compresión y descompresión, lo cual proporciona un mejor desempeño y calidad en las aplicaciones avanzadas, independientemente de la velocidad del microprocesador actual.

El software Indeo Video soporta, en la reproducción estándar, una resolución de 160x120 a 15 fps. Además permite un video acelerado de alta calidad y resolución de hasta 320x240 a 30 fps utilizando productos basados en el i750.

El procesador de pixeles 87750PB (Pixel Processor) cuenta con velocidad de video programable que soporta un amplio intervalo de compresión y otros algoritmos de video gráficas, mediante una RAM de instrucciones micromodificadas en el mismo chip. Este procesador de displays lee mapas de bits de una RAM de video, proporciona una descompresión final de mapas de bits YUv, y para un costo más bajo del sistema, incluye una tabla look-up de colores y un triple DAC (Digital /Analog Converter) de 8 bits.

El controlador de interrupciones 82489DX es el primero que soporta sistemas modernos y de multiproceso con el *mínimo tiempo arriba* (overhead), y tiene modos de operación compatibles con DOS y Windows.

El Pentium se encuentra disponible en las versiones de 66 y 60 MHz, la diferencia de rendimiento entre ambas versiones es de aproximadamente del 10%.

II.6 ARQUITECTURAS

* Comparación de Arquitecturas de UCP

Nuevos y rápidos procesadores están apareciendo en los sistemas de computadora personal. Los procesadores tipo CISC (juego complejo de instrucciones de computación) han dominado los diseños de PC's actualmente, mismos que son diseñados por las compañías Intel y Motorola; además se tienen los procesadores tipo RISC (juego reducido de instrucciones de computación), los que predominan en las estaciones de trabajo.

El nuevo procesador Pentium de Intel diseñado a partir de CISC, puede tener competidores muy fuertes con los chips del tipo RISC de mayor potencia.

* RISC Vs. CISC

Los procesadores RISC ofrecen un mejor manejo de instrucciones además de incluir un simple juego de instrucciones (128), mientras que los CISC utilizan un grupo de instrucciones más complejo (200 o 300). Los RISC utilizan un menor número de formatos de instrucciones al igual que menos modos de acceso a memoria que los CISC, resultando un hardware de control simple. El CISC produce una operación interna más lenta y ocupa espacio adicional en el chip para descifrar las instrucciones y la lógica de control que podría ser utilizada para optimización del rendimiento como cachés y registros mayores. El RISC permite implementar "*tuberías*" de instrucciones de múltiples etapas en el chip, con mayor eficiencia que los procesadores CISC. Una ventaja más del RISC sobre el CISC es que el primero incluye registros adicionales para realizar cálculos.

Una desventaja del RISC sobre el CISC es que a pesar de tener instrucciones más simples generalmente requieren más instrucciones para llevar a cabo las tareas.

* Intel Vs. Motorola

Los procesadores CISC más utilizados son el Intel 80X86 y el Motorola 680X0. Intel y los procesadores compatibles (AMD, Cyryx, IBM y Ti) se utilizan en las PC's, en tanto que los chips de Motorola se utilizan en las Machintosh de Apple. El procesador más popular en la PC es el 80486 y en Machintosh el 68040, ambos son de 32 bits con unidades de números reales de administración de memoria y 8 K de caché en el chip.

El 80486 mezcla cachés de instrucciones y datos, mientras que el 68040 presenta 4K de instrucciones y 4K de datos, además de unidades de administración de memoria dobles, lo

cual es una ventaja en las operaciones de manejo de instrucciones, ya que el acceso a éstas y a diferentes tipos de datos se debe realizar al mismo tiempo.

Actualmente Intel con su chip 486DX2 de duplicación interna de velocidad puede trabajar a 66 MHz, en tanto que el 68040, el más rápido de Motorola, tiene una velocidad de 33 MHz, por lo que las ventajas mostradas por la arquitectura Motorola se ven disminuidas con la mayor velocidad de Intel.

El Pentium muestra cambios notables en la arquitectura de los diseños estándares CISC de Intel. Continúa con diseño interno de 32 bits (registros, operaciones de enteros y bus de 32 bits). El Pentium incluye características encontradas en los procesadores RISC: el caché interno doble de 8K de instrucciones y 8K de datos; el bus de datos de 64 bits; velocidad de operación de 66 MHz. Mejor unidad de números reales y diseño superescalar, aumentan las ventajas de Intel sobre el 68040 de Motorola.

*** Nuevos Chips Tipo RISC**

Alpha

Digital Equipment Corporation en 1992 lanzó al mercado su chip Alpha, planeando una duración de su arquitectura de 25 años. Alpha está diseñado para apoyar a OSF/1, VMS y Windows NT. Este novedoso chip tiene una velocidad de 200 MHz, utilizando 1.68 millones de transistores. La velocidad obtenida en el chip se debe a la tecnología CMOS-4 (de Digital) de 0.75 micrones, operando a 3.3 volts.

Alpha es un diseño de 64 bits de emisión doble superescalar con un gran manejo de datos, contiene siete etapas de manejo de enteros y utiliza 8K de caché para instrucciones y 8K de caché para datos. La capacidad de emisión doble del Alpha se encuentra un poco limitada por lo que no puede realizar dos operaciones de enteros como el Pentium. Alpha puede realizar una instrucción de enteros y una de números reales, y algunas combinaciones de operaciones populares.

MicroSPARC y SuperSPARC

Sun Microsystems y Texas Instruments se unieron para fabricar los procesadores RISC SuperSPARC y MicroSPARC. El MicroSPARC es un procesador de una sola vía de acceso de 32 bits, que se utiliza en la SPARCstation Classic y la SPARCstationLX de Sun. El chip contiene 800 000 transistores con un caché doble (4K de instrucciones y 2K de datos), control de DRAM y la lógica de interfaz con el bus. Este chip se fabrica con un proceso CMOS de 0.8 micrones y alimentación de 5 volts, consumiendo 3.5 W a 50 MHz.

SuperSPARC es la primera versión superescalar de los procesadores SPARC implementando un diseño superescalar de tres emisiones que opera a 40 MHz, fabricado con un proceso BiCMOS de 0.8 micrones. Este procesador es de 32 bits con 3 millones de transistores, pero con características superescalares avanzadas que además de manejar tres instrucciones a la vez, puede manejar la dependencia de los datos entre las instrucciones en su unidad de ejecución.

Hewlett Packard Precision Architecture (PA/RISC) 7100

La arquitectura PA/RISC de Hewlett Packard realiza una gran ejecución de cálculos de números reales. El 7100 es un procesador de 32 bits que trabaja a 100 MHz. Se fabrica con un proceso CMOS de 0.8 micrones, conteniendo 850 000 transistores e incorporando una unidad de números reales en lugar de utilizar una unidad externa como se realizaban en las arquitecturas PA/RISC anteriores a ésta.

No se incluyen cachés en el chip, pero HP utiliza rápidos cachés de SRAM fuera del chip con acceso de 64 bits al procesador. Este sistema permite que se puedan incorporar cachés mayores o menores según se necesite; el caché de datos varía entre 4K y 2MB, mientras que el caché de instrucciones varía de 4K a 1MB. El 7100 es un diseño superescalar de dos vías de acceso (igual que el Alpha), también tiene la restricción de no poder emitir dos operaciones de enteros simultáneamente. HP tiene versiones de 33, 50 y 99 MHz del 7100.

IBM RS/6000 y Power PC

A principios de 1990, IBM introdujo su juego de chips RS/6000 basados en RISC, de 32 bits. El diseño del RS/6000 no es un sólo chip, sino que se utilizan siete o nueve chips separados según la operación, conteniendo 7.4 millones de transistores. El RS/6000 utiliza un diseño avanzado superescalar que permite realizar cuatro instrucciones si las condiciones lo permiten. Utiliza cuatro cachés de datos de 16K y un caché de instrucciones de 8K. IBM también ofrece una versión en la que se utiliza un solo chip del RS/6000 llamado RSC (RIOS Single Chip).

La arquitectura Power PC es el resultado de la unión de Apple, IBM y Motorola para la creación de un sólo chip de la arquitectura RS/6000. Esta arquitectura permite la compatibilidad entre Apple e IBM en términos de software. Los nuevos sistemas de Machintosh basados en el chip Power PC podrán ejecutar los programas del RS/6000 además de los programas 68000 binarios con la ayuda de emulaciones. Por su parte, las estaciones Power PC de IBM ejecutarán el software RS/6000 además de los programas binarios de la Machintosh. Adicionalmente, entre IBM y Apple están crendo el sistema operativo orientado a objetos que se ejecutará en los sistemas Power PC llamado Pink.

En octubre de 1992 se anunció el primer Power PC llamado 601. Utiliza la tecnología de interfaz de buses del procesador RISC 88110 de Motorola y la lógica del núcleo RSC de IBM. Se fabricó con un proceso de CMOS de 0.6 micrones y con 2.8 millones de transistores, de 32 bits operando a 50 o a 66 MHz. Este chip utiliza un caché unificado de 32K de instrucciones y datos, algo fuera de lo normal en los diseños RISC. Se alimenta con 3.6 volts y 9 W a 50 MHz.

- **Tarjeta Madre (SETUP)**

Las tarjetas 80286, 80386 y 80486 tienen una rutina de SETUP (programa utilizado para configurar la tarjeta madre) que puede ser guardado dentro de su CMOS RAM. Esta utilidad es usada para el programa de CMOS RAM con una lista de hardware que el sistema de la computadora ha instalado en ella.

La CMOS RAM almacena la siguiente información:

a) Fecha y tiempo.
b) Tipos de floppy y discos duros.
c) Características de UCP.
d) Tamaño de memoria base y extendida.
e) Tipo de video adaptador instalado.

Diferentes marcas de BIOS pueden también tener características más extendidas o avanzadas, que no pueden ser encontradas en el BIOS original del IBM AT.

El CMOS RAM puede ser reconfigurado (solicitando el SETUP), cuando se está:

a) Instalando el sistema por primera vez.
b) Cambiando los drives, tamaño de la memoria o adaptador de video.
c) Cambiando las características del sistemas (localización de la memoria RAM, etc.).
d) Reemplazando la batería de CMOS.

El ROM BIOS (Basic Input Output System) es un programa en código de máquina que controla a la computadora y la comunicación con el hardware. El BIOS interpreta comandos de programas comerciales que son ejecutados en memoria, los comandos son convertidos en comandos de lenguaje máquina y los manda a la UCP (808X y 80X86). La mayoría del software escrito realiza llamadas directas al ROM BIOS, estos son *llamados "programas de buen comportamiento"*. Los programas que se desvían del ROM BIOS y se comunican directamente con la computadora son conocidos como programas de mal comportamiento. Muchas tarjetas madre clones tienen su código de ROM BIOS localizada en EPROM's.

Las tarjetas madre que contienen un código de BIOS programado cumplen con las siguientes condiciones:

- Ejecuta el POST (Power On Self Test, autoprueba de arranque) el cual examina los componentes de hardware que se encuentran instalados en el sistema.
- Realiza una conexión entre el software y el hardware. Un programa de software solamente necesita hacer llamadas al ROM BIOS, sin que el hardware necesite llamarlo por sí mismo.

Un ROM BIOS compatible es visto por el software como si fuera exáctamente el IBM ROM BIOS para un sistema en particular, el código de compatibilidad del BIOS varía sólo en el área de la interfaz actual del hardware de la computadora.

Para ser una computadora con hardware compatible con sistema particular IBM debe cumplir con lo siguiente:

a) Debe utilizar el mismo canal IRQ del hardware y direcciones de puerto E/S hechas para el mismo propósito que IBM.
b) Ofrecer el mismo bus de interfaz que IBM. Usualmente con la misma velocidad de reloj.
c) Con un sistema AT verdaderamente compatible debe de funcionar igual que un IBM AT BIOS genuino.
d) Pequeñas diferencias que son aceptables, como son la velocidad del reloj y el número de estados de espera.

II.7 MEMORIAS

La memoria es la parte del sistema encargada de almacenar las instruciones y los datos en código binario. El elemento de almacenamiento de una memoria integrada comunmente empleado en microprocesadores, es el flip-flop tipo D. Tal flip-flop es cargado con el nivel lógico aplicado por la entrada de datos cuando recibe un pulso de reloj que lo habilita.

La necesidad de tener grandes capacidades de almacenamiento, tuvo como consecuencia la implementación de varios conjuntos de flip-flops en los circultos integrados. Actualmente, la matriz estándar de los flip-flops que conforman a una memoria, se distribuye de tal forma que se puede seleccionar una posición de la misma por medio de un decodificador. Ceda posición consta de 8 flip flops que permiten obtener una palabra de información de 8 bits cada vez que se realiza una lectura; esta información es depositada en las 8 líneas del bus de datos. Si se lleva a cabo una escritura en una posición de memoria, el contenido de las líneas del bus de datos se carga en los 8 flip flops de la posición selecionada por el decodificador. La figura 2.7 muestra la arquitectura típica de un chip de memoria con 256 posiciones de 8 bits cada una (256 x 8).

Además de los pines de alimentación de la señal de lectura/escritura y la de permiso de la figura mencionada, se tienen 8 pines más para activar las 8 entradas del decodificador y otros 8 para su conexión con el bus de datos. La conexión de la memoria al bus de datos de buffer triestado debe de quedar en alta impedancia cuando no se selecciona dicho elemento.

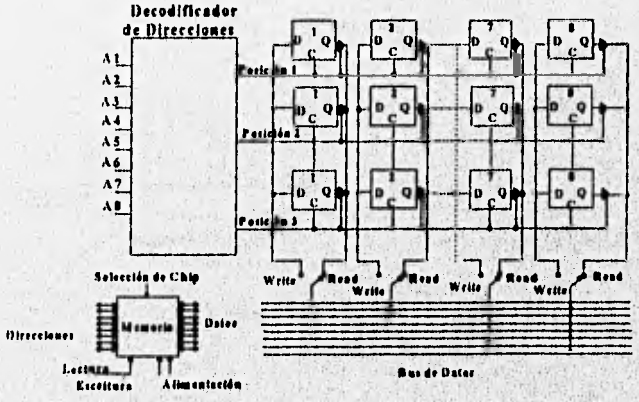


Figura 2.7 Arquitectura Típica de la Memoria 256

Continuando con la arquitectura de la memoria de la figura anterior, podemos decir que si aplicamos un nivel lógico cero a todas las entradas del decodificador a través del bus de

direcciones, se selecciona la posición 1 y el contenido de sus 8 flip flops pasaría por el buffer triestado hacia las líneas de bus de datos, si es que se trata de una lectura. Si se realiza una escritura, la información del bus de datos pasaría a los 8 flip flops de la posición seleccionada.

Existen dos tipos fundamentales de memorias:

- RAM (Random Acces Memory): Memoria de acceso aleatorio.
- ROM (Read Only Memory): Memoria de sólo lectura.

II.7.1 MEMORIAS RAM Y ROM

Las memorias RAM y ROM son de acceso aleatorio, por lo que se puede tener acceso a cualquier posición de ellas sin seguir un orden determinado.

Las localidades de las memorias RAM pueden ser leídas o escritas, en tanto que las memorias ROM sólo pueden leerse una vez que hayan sido grabadas.

Las memorias RAM pueden ser construidas a base de flip flops tipo D, pero también por medio de células o bloques dinámicos. Las memorias RAM que utilizan flip-flops, una vez que se ha escrito sobre de ellas, mantienen su contenido mientras permanezca la alimentación. Las células dinámicas de las memorias RAM se basan en la carga de un capacitor por medio de la conducción de un transistor de tipo MOS (Ver Fig. 2.8).

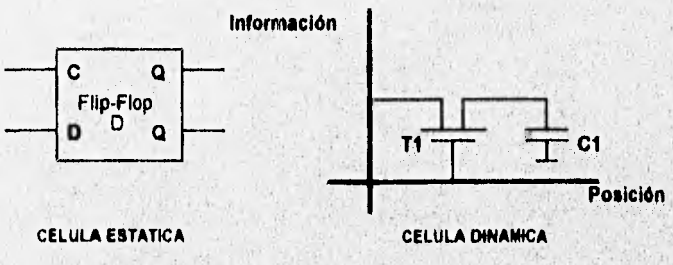


Figura 2.8 Células de las Memorias RAM

Las memorias RAM que utilizan células dinámicas, tienen las siguientes ventajas:

- Reducción en una tercera parte del costo por bit.
- Disipación de potencia cuatro veces menor que las células estáticas.
- Aumento cuatro veces el nivel de integración respecto a las células estáticas.
- Reducción del tiempo de acceso a más de la mitad del utilizado por las células estáticas.

Pero además de las ventajas anteriores, este tipo de memorias también tiene ciertas desventajas con respecto a las células estáticas:

- Necesidad de utilizar tres tensiones de alimentación contra una de las estáticas.
- Las fugas que tiene todo tipo de capacitor, exigiendo un *circuito de refresco*, que nos permita reescribir cada cierto tiempo la información, ya que el capacitor se descarga y pierde su contenido muy rápido. El circuito debe recargar todas las células en un intervalo de pocos milisegundos compensando de esta manera las pérdidas producidas por las fugas.

Posteriormente fueron diseñadas las memorias IRAM para suplir a las RAM estáticas, sin sus inconvenientes. En ellas se encuentra internamente una memoria RAM dinámica con los circuitos de refresco integrados en el mismo chip, por lo que tienen un comportamiento de RAM estática, pero sin sus desventajas; no son de fácil acceso al público debido a su costo.

Las memorias ROM se basan en *conexiones* en los cátodos de una matriz de diodos que implementan los bits de memoria. Al observar la figura 2.9 se puede apreciar que la conexión del cátodo de uno de los diodos que actúa como células elementales a la línea de información, representa un nivel 1 en esta última, al permitir la circulación de corriente por su correspondiente resistencia de absorción Ra.

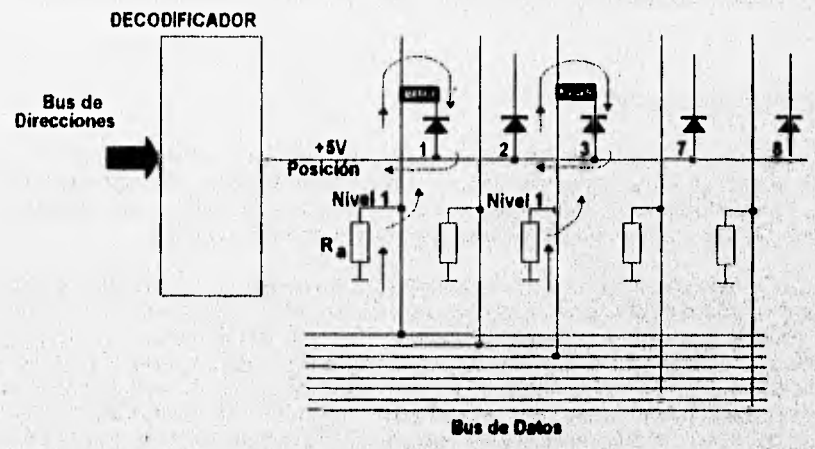


Figura 2.9 Diagrama de la Memoria ROM

Si el cátodo de un diodo no se conecta, no circula corriente por Ra por lo que no existirá tensión (nivel 0) en la línea de información correspondiente.

Según el tipo de conexión que se utilice, las memorias ROM se dividen en el siguiente grupo:

- **Memoria ROM de máscara**. Por medio de sofisticados procedimientos se diseña de fábrica una máscara que permite conectar el cátodo de los diodos que deben de contener el nivel 1. La elaboración de la máscara resulta ser muy cara por lo que sólo interesa cuando la producción alcance cifras superiores al millar de unidades.

- **Memorias PROM (Programable ROM).** El fabricante provee este tipo de memorias con un fusible especial que sirve de conexión. Mediante un *programador de PROM* se funden los fusibles correspondientes a los diodos que debe coincidir con un nivel lógico cero. Este tipo de memorias es ideal para pequeñas series y prototipos.

- **Memorias EPROM (Erasable PROM).** Son de utilización similar a las PROM, pero el proceso de grabación no es destructivo. Los datos son almacenados induciendo una carga eléctrica en unos electrodos de polisilicio, aislados completamente en una capa de SiO_2. La grabación se lleva a cabo por medio de un programador adecuado; en ese momento los datos permanecen aunque la alimentación haya sido eliminada; sin embargo, es posible borrar toda la información y reutilizar la memoria si ésta es expuesta a rayos ultravioleta. Una EPROM que ha sido borrada puede ser grabada con información diferente, pero esto sólo se puede realizar un número limitado de veces, ya que el estado interno de la memoria se degrada en cada borrado.

- **EEPROM (Electrically Erase ROM).** Es muy similar a las EPROM, con la diferencia de que éstas pueden ser borradas aplicando determinada tensión en sus terminales. Este tipo de memorias es muy utilizada en equipos que requieren reprogramarse en los mismos aparatos.

II.7.2 MEMORIA CACHE

La utilización de memoria caché permite que los procesadores realicen operaciones con mayor rapidez. La arquitectura caché de un microprocesador moderno, es similar a un vector de las supercomputadoras tradicionales, que usaban registros de este tipo para guardar los procesos utilizados sin tener frecuentemente referencia a la memoria principal.

Caché se basa en la observación de muchos programas de aplicación que exhiben algunos grados de localidad de referencia; programas que accesan piezas de datos que se encuentran listos para ser llamados en un espacio y tiempo. Un programa que accesa memoria sin respeto a la localidad de referencia puede ser ejecutado pobremente porque ha perdido un alto número de caché. El compilador juega un papel crucial en la reestructuración de programas para reducir las perdidas de caché por intercambio de información, o por el bloqueo del buffer, así que esos datos son ejecutados más eficientemente por el procesador. Este arreglo es similar a la vectorización original de compiladores que reestructuran programas justo en el vector de memoria (registros) en piezas. En otras palabras, el compilador estructura programas que son muy utilizados en un subconjunto de problemas que pueden estar justo dentro de la memoria caché, así el procesador puede trabajar sobre remiendos o parches del código original y datos provenientes de la memoria caché (además evitando hacer referencias a la memoria principal) antes de moverse sobre el siguiente parche.

II.7.3 MEMORIA FLASH

La tecnología ETOX TM de Intel, dada a conocer en 1988, permite la producción de memorias EPROM de lectura-escritura de alta densidad, conocidas como memorias flash, que

realmente son memorias RAM no-volátiles de alto desempeño. Un chip de memoria flash se caracteriza por tener un muy bajo consumo de potencia, ser compacto, robusto y de gran confiabilidad en su operación. La tendencia económica de este tipo de memorias es bajar de precio debido a:

- Economías de fabricación inherente en el proceso ETOX TM.
- Aumento en la densidad de memoria producible.
- Rápido crecimiento en el volúmen de producción.

La no volatibilidad y la velocidad de acceso similar a una RAM dinámica (DRAM), son dos características importantes de las memorias flash, que las hacen ideales en la construcción de discos duros. Un disco duro basado en memorias flash es mucho más rápido que los que no la utilizan, aproximadamente de 125 000 a 250 000 veces que un sistema mecánico. Por ejemplo, al utilizar la memoria flash, en 120 ns se puede tener acceso a los datos almacenados con una densidad de 1 a 40 MB, mientras que toma de 15 a 30 ms recuperar datos de un disco duro normal que a su vez puede almacenar desde 5 MB hasta 1 GB.

Las memorias flash se pueden encontrar disponibles en densidades de hasta 8 MB, pero si se utilizan tarjetas flash se pude tener disponibilidad de 4, 10, 20 y 40 MB con la ayuda de dispositivos Boot Block en densidades de 4 MB.

La memoria flash así como la EPROM, necesita una fuente de programación Vpp =12 V ±5 % además de su alimentación de 5 V. También existen memorias flash que operan y se programan con 5 V como fuente única del sistema.

*** Tarjetas de memoria**

La primera generación de las tarjetas de memoria flash tienen la opción de poder reescribirse eléctricamente, es no volátil, confiable, robusta y económica en altas densidades.

La segunda generación de tarjetas de memoria flash proporciona aún mayores densidades, menor consumo y mayor funcionalidad, debido a que soportan hasta 20 MB y un registro de control CMRS (Component Management Register Set) basándose en el chip 28F008SA. El control por software de muchas funciones de esta tarjeta se encuentra en un archivo de software interno.

Ejemplificando lo anterior, podemos mencionar que la M28F256 es una memoria flash a la que se le aplica una descarga eléctrica para ser borrada y reprogramada para un nuevo uso. Es ideal para hacer historiales de códigos o para tablas de datos que pueden ser controlados, por ejemplo, en la información de un periódico en el que es necesario hacer adaptaciones constantemente según sea requerido.

La necesidad de un código de datos ascendentes da paso a los sistemas de larga duración para crear prototipos de sistemas de manufactura y servicios. Mientras dura el prototipo se utiliza como tal; sin embargo, cuando es necesario cambiar los datos no se necesita utilizar un código de luz ultravioleta como en la EPROM. La memoria flash reemplaza los 10 o 15 minutos de exposición en luz ultravioleta por una descarga eléctrica de 1 segundo para ser borrada.

El material y la labor asociados al costo del cambio de códigos se incrementan para sistemas de integración. Utiliza componentes discretos.

Es una memoria alterable, elimina espacios de memoria, reduce material de alto costo y presenta una drástica reducción en los costos asociados al apilamiento de datos; es flexible y compatible con las EPROM Y EEPROM, pero además presenta la ventaja de poder ser reprogramadas día con día de acuerdo a las necesidades requeridas.

*** Acceso Directo a Memoria (DMA, Direct Memory Access)**

El *acceso directo a memoria* (DMA) es una técnica de entrada/salida (I/O) que desvía al microprocesador para transferencia de datos. Durante ésta, el microprocesador es aislado, es decir, el DMA proporciona acceso directo a la memoria mientras el microprocesador se encuentra en estado de espera (Standby), esto permite que los datos puedan ser transferidos entre la memoria y el sistema I/O con tasa de velocidad que es limitada solo por la rapidez de los componentes de memoria en un sistema. La velocidad del DMA puede ofrecer un acceso con velocidad de transferencia de datos de 8 a10 Mbytes por segundo con los componentes de memoria RAM de alta velocidad de hoy en día.

La transferencia de datos por medio del DMA es utilizado para muchos propósitos pero el más común se utiliza en el refresco de pantallas CRT y sistemas de lectura y escritura de discos. El DMA es utilizado también en algunas ocasiones para transferencias de datos a alta velocidad, de memoria a memoria o al sistema I/O.

En el modo mínimo del DMA se utiliza al 8086/8088 operando con dos señales de control necesarias para solicitar y admitir al DMA. Estas señales se encuentran en los pines HOLD (pin 31) y HLDA (pin 30) del 8088/8088. El pin HOLD solicita un trabajo del DMA mientras que el HLDA permite que éste realice el trabajo. Siempre que HOLD se encuentre activado, las direcciones, datos y control de buses del 8086/8088 estará convirtiéndose en tres estados, comenzando por el de alta impedancia. La única señal de control de bus que no se encuentra en estado de alta impedancia es la señal ALE (pin 25), la cual es un cero lógico durante un HOLD.

La señal de salida HLDA empieza a activarse para indicar que el 8086/8088 tiene localizados los buses en estado de alta impedancia.

La entrada HOLD para el 8086/8088 requiere sincronización externa con el reloj, para lo cual, un flip-flop tipo D es conectado en el pin HOLD y la llamada externa de la señal.

*** Definiciones Básicas del DMA**

Las operaciones del DMA ocurren normalmente entre el sistema de Entrada/Salida (I/O) y la memoria. Una lectura del DMA transfiere datos de la memoria (lectura de memoria) hacia el sistema de I/O. Una escritura del DMA transfiere datos del sistema I/O a la memoria. En ambos casos, la memoria y el sistema de I/O tienen que ser controlados simultáneamente.

Un ciclo de verificación envía las direcciones del DMA a la memoria pero no realiza una lectura o escritura para cualquiera de los sistemas de ésta o de I/O.

El bus del 8086/8088 tiene señales de control que no se encuentran disponibles en el control del sistema del DMA. Para corregir esta situación, los tres controles se han convertido en cuatro señales de control para el DMA basado en el sistema de I/O.

Un controlador de DMA es un microprocesador de propósito especial diseñado para transferir datos a través del bus de datos entre la I/O y la memoria. El controlador básico del DMA contiene un registro de direcciones y un contador para verificar el número de bytes transferidos.

El controlador DMA programable 8237A-5 es un sistema de cuatro canales capaz de transferir datos arriba de los 64 Kbytes, usando técnicas DMA sin la intervención del 8086/8088. Se pueden transferir datos con este controlador a velocidades cercanas a los 1.6 Mbytes por segundo cuando opera con un reloj de 5 MHz.

El controlador DMA 8237A-5 opera en cuatro modos básicos:

i) *Modo de demanda*. Este modo puede transferir datos tan largos como la petición de entrada para un canal que está activo.
ii) *Modo sencillo*. Puede transferir un byte cada vez que la llamada de petición se encuentre activa.
iii) *Modo bloque*. Permite transferencia de datos a la memoria por medio de software.
iv) *Modo cascada*. Expande al sistema más de cuatro canales que originalmente eran utilizados.

Las direcciones y el contador de registros del 8237A-5 son de 16 bits de ancho, lo que permite al controlador direccionar los datos a unos 16 bits de memoria como límite y transferir arriba de 64 Kbytes de datos con una programación.

El 8237A-5 trabaja adecuadamente con el 8086/8088 por medio de un latch externo de direcciones que captura los bits A_{15}-A_{18}, y un latch adicional que permite almacenar 4 bits de dirección de memoria alta (A_{19}-A_{16}).

El bus local es el que está residente en el microprocesador, y sobre él están la memoria y los puertos de I/O, a los cuales el microprocesador tiene acceso. Esta memoria y la I/O local son únicamente accesados por el microprocesador que se encuentra conectado directamente sobre ellos. Un microprocesador operando en el modo de bus local puede contener sus propios datos, direcciones y bus de control, los cuales son conectados al bus remoto. Este es compartido, siendo accesible para todos los microprocesadores en un sistema. El bus remoto puede tener su propia memoria y la I/O, y a su vez son compartidos entre todos los microprocesadores de un sistema, en tanto que el bus local está residente para uno solo.

El término de *bus maestro* se aplica para cualquier sistema (microprocesador o cualquier otro) que pueda controlar un bus conteniendo memoria y la I/O. Si se requiere el control del acceso al bus compartido, el bus maestro tiene que pedirlo a través del bus árbitro. El *bus árbitro* es un sistema que controla el acceso al bus remoto, esto también resuelve la prioridad cuando más de un bus maestro piden acceso al bus compartido.

El circuito 8289 es un bus árbitro que se conecta a microprocesadores con sistemas de bus compartidos, esto permite al bus maestro operar en tres diferentes modos:

a) *Bus en modo sencillo*, el cual permite a un microprocesador tener acceso al bus compartido.

b) *Bus en modo I/O*, que permite a un microprocesador tener acceso a un bus compartido y a un bus local I/O.

c) *Bus en modo residente*, que permite al microprocesador accesar a un bus compartido, y a un bus local I/O y de memoria.

*** Errores en Memoria**

La memoria en la tarjeta madre y en la tarjeta de expansión se encuentra organizada en bancos de chips. Las tarjetas de memoria expandida utilizan una arquitectura de 8 bits y sus bancos utilizan 8 bits de ancho. Las tarjetas de memoria extendida usan una arquitectura de 16 bits y sus bancos utilizan 16 bits de ancho. Con paridad en memoria existe un bit adicional por cada 8 o 16 bits. Un banco que tiene 9 chips de RAM utiliza 8 chips para la memoria actual, y el noveno es utilizado para checar la paridad.

Muchos de los nuevos sistemas utilizan chips SIMM (Single Inline Memory Module, Módulo de Memoria Simple en Línea) que reemplazan los viejos DIP (Dual Inline Package, Paquete en Línea Doble). Cada SIMM contiene bytes enteros de memoria o bien bits individuales, por lo que es fácil determinar qué banco se encuentra dañado.

Para entender las posibles fallas que podemos encontrar en las memorias, es importante considerar los siguientes conceptos:

- ***Tiempo de acceso***, es el tiempo medido de datos que llegan a las direcciones de entrada de los chips de memoria para el tiempo de datos estable a su salida. Este tiempo es medido en nano segundos (ns) y normalmente está etiquetado en el chip RAM. Muchos usuarios tienen la idea de que instalar chips RAM veloces incrementarán la velocidad de la computadora, lo cual no es cierto. La velocidad de entrada y salida de datos de la RAM es fijada por el sistema de velocidad de reloj de la computadora. El único beneficio que se puede obtener al instalar chips más rápidos es una menor probabilidad de encontrar errores de paridad. Si se desea realmente cambiar la velocidad de la computadora, se debe cambiar el cristal de la tarjeta madre y posiblemente también sea necesario cambiar la UCP, pero antes de realizar la operación anterior se debe verificar si la tarjeta madre soporta dichos cambios.

- ***Banco de switches***, implica el mapeo de un conjunto de memoria expandida para una dirección vacía, abajo de 1024Kb.

- ***DRAM (Dynamic Random Acces Memory)***, los datos son almacenados como cargas eléctricas que se refrescan constantemente. Este tipo de chips tienen un ciclo o secuencia de señales de control cada vez que la memoria es accesada.

- ***Memoria Extendida***, es una memoria lineal que inicia en direcciones arriba de 1MB (FFFF h). La operación en modo real requerido por DOS no soporta el acceso a memoria extendida. Si el usuario necesita transferir datos para y de la memoria extendida, la UCP necesita un switch de modo real para proteger el modo de operación. Este tipo de memoria se encuentra sólamente desde el microprocesador 80286 en adelante.

- ***Memoria Expandida***, es una memoria no linear mucho mayor de 1MB a la cual se puede acceder por medio de una base giratoria en bloques; esto es posible por medio de un Manejador de Memoria Expandida, direccionado dentro de DOS con un límite de 1MB. La

memoria expandida sólo pueder ser accesada por medio de un banco de switches, el cual provee una pequeña ventana a través de la memoria, y los bloques de la memoria expandida son manejados con la memoria base.

- ***Manejador de Memoria Expandida (EMS)***, no se puede tener más de un EMS. El EMS es un software instalado en la memoria de la computadora al mismo tiempo que se iinicializa la computadora.

- ***Compuerta A20***. La UCP que utiliza el microprocesador 80286 puede conmutar del modo real al modo protegido de operación, pero no a la inversa; tal conmutación es realizada por la compuerta A20 del microprocesador. El 80386 tiene una instrucción que conmuta de un modo a otro adecuadamente. Para que pueda llevarse a cabo la conmutación del modo protegido al real de operación, la UCP necesita una señal de nivel alto en el pin de reset de la misma para ser reinicializada.

- ***Página de memoria***. La memoria se encuentra dividida en bloques de 16K que son llamados páginas. Estas pueden ser reubicadas desde la memoria expandida a la memoria convencional por medio del banco de switches.

- ***Modo protegido***, es un modo especial de operación que permite direccionar arriba de 16MB de memoria extendida en sistemas 80286 y posteriores. Éste es el modo nativo de los procesadores 80286 y 80386, y permite a la UCP ejecutarlo como fue diseñado, con acceso total a toda la capacidad de memoria. Este modo operacional es incompatible con el software escrito para modo real de operación. Múltiples operaciones pueden ejecutarse concurrentemente mientras que la integridad de cada aplicación y sus datos pueden ser protegidas de otras. Un *"selector"* es simplemente una compensación dentro de una tabla de descriptores. Un *"descriptor"* contiene la dirección base y longitud de un segmento.

El *Modo real* también se refiere al modo de compatibilidad entre los chips 8086 y 8088. Cuando el procesador 80386 se encuentra en modo real, la emulación de la UCP 8086 es completada, incluyendo cualquier limitación que el chip 8086 tenga.

- ***Modo virtual 8086***. Pertenece a los procesadores 80386, 80386SX y posteriores. Este modo permite al usuario ejecutar varias sesiones de modo protegido como una o más subtareas, y emula un programa 8086 completo. Cada modo virtual require 1 MB de memoria.

- ***Estado de espera***. El número de estados de espera indica cuantos ciclos de reloj debe de esperar la UCP a la memoria para tener datos disponibles.

*** Tipos de prueba de memoria**

Para llevar a cabo una revisión sobre el estado de la memoria se tienen las siguientes pruebas de:

a) Almacenamiento y lectura.

b) Número secuencial.
c) Bits en rotación.
d) Paso a paso de bits.
e) Direcciones dobles.
f) *Butterfield.*
g) Suma.

Los problemas con tarjetas de memoria pueden ser causadas por:
1) Conflicto con las direcciones de E/S y otra tarjeta.
2) Un chip en mal estado sobre la tarjeta.
3) La tarjeta no se encuentra instalada correctamente en la ranura de expansión.
4) El software no se encuentra configurado correctamente.
5) Soldaduras en mal estado u oxidación en los contactos.
6) Alguno de los chips no se encuentra bien orientado en su socket.
7) Los switches de configuración no se encuentran en el lugar correcto.
8) El manejador de memoria no se ha instalado o la versión no es la adecuada.
9) El manejador de memoria se encuentra en conflicto con otro manejador.
10) Los chips de RAM son demasiado lentos para la velocidad del reloj.
11) La computadora no reconoce la memoria. Cuando esto sucede debe hacerse lo siguiente:

- Checar el conjunto de switches de las tarjetas de memoria y la tarjeta madre, y ejecutar el programa de SETUP para sistemas de tipo AT.
- Checar el conjunto de switches de las tarjetas de memoria y tarjeta madre de los sistemas tipo XT.

II.8 DISPOSITIVOS DE ENTRADA/SALIDA

II.8.1 MONITORES

*** Conceptos Básicos:**

Tecnología: CTR (*Tubo de rayos catódicos*). LCD (*Cristal líquido monocromático*). LCDC (*Cristal líquido color*). TFT (*Thin Film Transistor*).

Deflexión: Expresado en valores sexagesimales, su valor indica si se trata de una pantalla más o menos plana. Cuanto más plana sea la superficie frontal del tubo (mayor grado de deflexión) menor deformación sufrirá la imagen en los bordes de la misma.

Cromaticidad: Si el monitor es B/N (blanco y negro) o Color.

Resolución Máxima, Modo Entrelazado: Si figura la palabra 'NO' es que el monitor no puede trabajar en modo entrelazado a ninguna resolución.

Resolución máxima, modo no Entrelazado: Esta es la máxima resolución a la que puede trabajar el monitor en modo de video. Si figura un 'NO', es que todas las resoluciones del monitor se presentan en modo entrelazado.

Compatibilidad: Se indican las diferentes compatibilidades con los estándares del mercado: EGA, VGA, Super VGA, 851/A, XGA; integración del monitor en el equipo y el aprovechamiento o no de sus posibilidades.

Tamaño de punto: Parámetro que define la calidad de imagen ya que está estrechamente relacionado con la resolución. Expresado en mm, es el diámetro de los orificios de la rejilla a través de la cual pasan los rayos catódicos generados por el monitor. Cuánto menor sea el diámetro, mayor será la densidad de puntos mejorando la calidad de la imagen.

Señal de entrada: Sistema de transmisión de la señal de video desde la computadora a la pantalla; puede ser video compuesto, analógico o digital.

Multifrecuencia: Un monitor Multifrecuencia o MultiSync es aquel capaz de adaptarse automática o manualmente a diversas frecuencias de señal de video, generadas por diferentes tarjetas gráficas existentes en el mercado o dentro de una misma tarjeta a sus distintas resoluciones. Se indican los valores mínimo y máximo de la frecuencia de sincronización vertical en Hertz. Si no es multifrecuencia se indica con un 'NO'.

Baja radiación: Indica si la emisión de radiaciones electrostáticas y electromagnéticas es baja o no, con el fin de evitar riesgos potenciales para la salud. Una reducción en la acumulación de carga estática elimina el riesgo de descargas eléctricas y reduce significativamente el polvo adherido a la pantalla.

Altavoz incorporado: Es un altavoz con o sin control de volumen, haciéndo posible que el monitor emita los sonidos procedentes de la computadora.

Control digital de imagen: En los monitores multifrecuencia, cuando varía el modo gráfico, los sincronismos y la frecuencia de barrido suelen modificarse simultáneamente, por lo que también se altera en mayor o menor grado la posición de imagen en la pantalla. Este sistema de control se encarga de centrar y ajustar automáticamente la imagen de modo de resolución.

*** Monitores TFT LCD para Notebooks**

Las pantallas LCD ocupan el segundo lugar en el mercado de ventas de display, sólo después de los CRT. Los LCD han mejorado el desempeño del display gracias a los avances e innovaciones técnicas en la fabricación de estos y sus principales componentes. Los monitores TFT LCD (*Thin Film Transistor Liquid Crystal Display*) que usan un dispositivo de control por cada pixel (o subpixel), ofrecen una respuesta rápida y excelente desempeño de display comparados con los CRT. Como consecuencia, los TFT LCD tienen un papel importante entre los display o pantallas planas.

Actualmente existe una gran variedad de TFT LCD, cuyos tamaños varían desde 3 (para receptores de TV portátiles) hasta de 17 pulgadas (utilizado en estaciones de trabajo). La tendencia del uso de los TFT LCD debido al desarrollo tecnológico se dirige principalmente a las computadoras notebooks.

Existen cuatro categorías de clasificación para los TFT LCD dependiendo del número de pixeles necesarios para su uso en las computadoras personales y en las estaciones de trabajo

en ingeniería. En el primer grupo están los modelos que producen 640x480 pixeles para sistemas compatibles VGA. En un segundo grupo los de 800x600 pixeles que se emplean en los equipos compatibles con Super VGA. En tercer plano se encuentran las pantallas que producen 1,024x768 pixeles para equipos compatibles con XVGA. Por último, en cuarto lugar se tienen los TFT LCD con 1,280x1,024 pixeles para equipos compatibles con Super XVGA.

Los TFT LCD para VGA compatibles proporcionan 512 colores, trabajan con pixeles que van de 0.3 a 0.33 mm y miden de 24 a 26 cm (9.4 o 10.4 pulgadas) a través de la diagonal. Estos paneles muestran ocho escalas de grises cada uno para los colores RGB (red, green, blue), y funcionan con dos lámparas fluorescentes de cátado frío.

Los TFT LCD para computadoras subnotebooks son objeto de una gran demanda. Estos paneles ultra compactos de diseño ligero, emplean pixeles de 0.25 mm y miden 20 cm (7.8 pulgadas) diagonalmente.

La forma en que se clasifican los tamaños de los monitores puede dar una apariencia errónea. Los tamaños se dan como medidas diagonales, de esquina a esquina del CRT (Tubo de Rayos Catódicos) en lugar del cristal visible dentro del marco del monitor. Por ejemplo, en los monitores de 14 pulgadas, el tamaño diagonal medido del cristal dentro del marco, desde la esquina inferior izquierda diagonalmente hasta la superior derecha, incluyendo cualquier borde negro, puede ser desde 13 hasta 13.8 pulgadas.

La opción más utilizada de monitor a color en nuestros días es el VGA de 14 pulgadas, pero los monitores de multifrecuencia VGA, SuperVGA (800x600) y algunas veces resoluciones de 1,024x768 o mayores están renovando al anterior VGA. La actual renovación de monitores se debe a que un monitor SVGA sólo consume dos terceras partes del área de pantalla que tomaría un VGA.

Los pixeles en cada caracter están más cercanos en una resolución alta que en una baja, la resolución alta resulta en caracteres mejor formados, por lo que el tamaño de la pantalla se hace más importante en las resoluciones más altas. En cualquiera de los tamaños de pantalla los caracteres son más pequeños en el modo SVGA que en el VGA.

*** Controladores de CRT**

El sistema conocido como *CRTC* (Cathodic Ray Tube Controler) se encarga de generar una señal de video que en una pantalla de adecuada resolución despliegue una imagen lo más perfecta posible.

Actualmente con el desarrollo de nuevos semiconductores, se han diseñado circuitos controladores de video con velocidades y capacidades de trabajo mayor que sus antecesores. Estos controladores realizan funciones complejas como el control de la memoria de video, el control de la pantalla o generación directa de formas geométricas. Estos controladores tienen una supervisión continua por la UCP, pero tienen capacidad de elaboración propia que libera a la computadora principal de las operaciones gráficas que se debían de realizar previamente.

Un sistema de video puede dividirse en dos partes:

- Generación de lista de video.
- Control de video.

Un procesador (regularmente la UCP) o controlador especifico, convierte las instrucciones del programa en instrucciones para formar figuras geométricas básicas. Este controlador o UCP, genera como salida lo que se conoce como *lista de video*, la cual es una serie de instrucciones gráficas que se envían al controlador de video.

En la memoria se modela cualquier figura a nivel de programa, que es transmitido al controlador de video encargándose de convertirlo en una señal de video y dar forma al objeto
modelado. Dicho controlador de video ejecuta la lista de instrucciones escribiendo la información a cada pixel en la memoria de la pantalla. El CRTC o circuitos específicos, se encargan de leer la memoria y generar la señal de video correspondiente a la información.

Las operaciones de modificación del contenido de la memoria de video generan una serie de operaciones tales como crear una nueva dirección; leer la memoria de pantalla, modificar el contenido y volver a escribir los datos ya modificados. Los procedimientos anteriores hacen lento el trabajo realizado por la UCP debido a la capacidad de elaboración que requieren de ésta, y que no puede dedicarse a otras operaciones. Para evitar lo anterior, las nuevas generaciones de controladores gráficos han utilizado un método de generación de texto y gráficos conocido como *Bit Boundary Block Transfer* (BiBlt o Raster), el cual hace que el procesador trabaje en términos de números de bit por pixel. Lo anterior trae como consecuencia la utilización de mayor memoria en imágenes a color y de alta resolución, por lo que se asegura un rápido movimiento de las zonas completas de la memoria de video o zonas de pantalla.

*** La memoria de Video**

La memoria de video se encuentra formada por una memoria RAM que combina las ventajas de la densidad de una DRAM con la capacidad de desplazamiento de registros, dando facilidades en el diseño de *buffers de cuadro* de gran ancho de banda y de mínimos componentes. En diversas ocasiones y para obtener mayor velocidad de respuesta, la memoria de video es una específica y tiene dos puertos de acceso, en uno de ellos se leen los datos para ser representados en pantalla, mientras que por el otro puerto se modifican los datos.

Los controladores modernos tienen la capacidad de direccionar 4 MBytes, por lo que no toda la memoria se utiliza para la representación simultánea en pantalla por medio del plano de bits, y se pueden transferir bloques de pixeles al plano de bits desde otras posiciones del plano de pantalla. Algunos controladores utilizan la memoria de pantalla para memorizar elementos no gráficos, como instrucciones, tables, datos, etc.

Al representar imágenes en blanco y negro sólo se modifica un bit del plano de bits por cada pixel, pero para desplegar imagenes a color o escalas de grises se tienen ciertos problemas, ya que se requiere de un número de bits dependiente del número de colores que

se pueden representar simultaneamente en la pantalla. En los casos anteriores, la modificación de un pixel significa que debemos tener acceso al plano o planos de bits que deseen modificar, a la palabra que los contiene y realizar la operación lectura-modificación-escritura. Todo el proceso anterior tiene que llevarse a cabo en menos tiempo de lo que dura un refresco de pantalla, ya que si no es así, se podría tener una inadecuada operación y los colores que se querían desplegar en la pantalla se cambiarán por otros mientras se lleva a cabo la operación.

En el caso de las computadoras personales IBM o compatibles, las resoluciones de video se han clasificado de la siguiente forma (Tabla 2.1):

Nombre	Adaptador de Video	Resolución	Colores
HGA	Hercules Graphics Adapter	720 x 384	2
CGA	Color Graphics Adapter	320 x 200	4
EGA	Enhanced Graphics Adapter	320 x 200	16
		640 x 200	16
		640 x 350	4 o 16
		640 x 350	2
VGA	Video Graphics Array	640 x 480	2
		640 x 480	16
SVGA	Super Graphics Video Array	800 x 600	2
		800 x 600	16
		800 x 600	256
		1024 x 768	256

Tabla 2.1 Resoluciones de Video

*** Errores en Monitores y Tarjetas de Video**

Los problemas con tarjetas de video y monitores pueden ser causados por distintas razones:

1) El switch en la tarjeta madre no se ha seleccionado adecuadamente. En sistemas PC-XT puede elegirse entre los modos monocromático y color. El sistema de AT tiene un switch o jumper, pero también se utiliza el SETUP, que permite al CMOS conocer el tipo de adaptador que ha sido instalado.

2) El software no soporta la manufactura o modo de la tarjeta de video. Si la configuración del software no presenta la especificación de un monitor EGA, VGA o PS/2, y se está utilizando un monitor a color analógico, se puede configurar el SETUP del programa en el modo *color* si éste lo contiene. Si el software no soporta el modo VGA y necesita un modo CGA o Hércules, se debe de reconfigurar la tarjeta VGA por medio de jumpers (si es esto posible) o ejecutando una utilidad de software especial que emula el modo de video que se necesite.

3) Los comandos de DOS, tal como MODE CO80, no funciona en modo extendido. Se debe configurar el monitor en modo estándar y volver a intentar con el comando.

4) La tarjeta adaptadora puede no estar bien colocada en la tarjeta de expansión; uniones de soldadura o algún chip en mal estado. Alguno de los chips de la tarjeta no se encuentra colocado u orientado correctamente en su socket. El cable de la tarjeta al monitor está desconectado.

5) El manejador (driver) de adaptador se encuentra mal o está en conflicto con otro.

6) El puerto paralelo de la tarjeta de video no funciona. Se debe de verificar si no existe problema con otra tarjeta adaptadora utilizando la misma dirección de puerto. Para verificar si el puerto es el problema se tiene que realizar una prueba de Loop-Back. Esta prueba consiste en enviar y recibir caracteres en el puerto al mismo tiempo, con la ayuda de un conector especial que se utiliza para llevarla a cabo (ver capítulo V).

7) El software de gráficas no es capaz de ser ejecutado en un adaptador monocromático de gráficos. Se debe verificar si se puede realizar una configuración con jumpers para soportar modo gráfico. Si no existen jumpers se puede utilizar algún programa para habilitar las páginas de gráficos.

8) Las tarjetas monocromáticas de gráficos MGC o MGA, CGA o EGA instaladas al mismo tiempo en el sistema pueden causar problemas, y esto puede provocar un error y desplegarse en pantalla el siguiente mensaje:

PARITY CHECK 1
?????

Si adaptadores MGA y EGA son instalados al mismo tiempo, entonces en la segunda página de video se escriben aleatoriamente caracteres y colores que pueden ser desplegados en la pantalla del monitor EGA, en este caso, el sistema puede no tener problemas. Para usar ambas tarjetas al mismo tiempo se necesita ejecutar la tarjeta monográfica en modo medio (HALF MODE) para evitar problema de mapeo.

9) Dirección E/S de la tarjeta EGA está mal configurada.

10) Cuando se realizan cambios de modos de video entre los adaptadores de EGA o VGA conectados a un monitor multifrecuencia, la imagen puede cambiar. Para tratar de mejorar ésta, se tienen que ajustar los controles horizontal y vertical hasta que la imagen se centre en una posición correcta. Algunos monitores multifrecuencia no sincronizan apropiadamente las señales que son enviadas por la tarjeta de video. El monitor puede no ser capaz de sincronizar la entrada correctamente. Si la imagen tiembla después de algún tiempo, es probable que la tarjeta de video no sea compatible.

11) La imagen toma algunos segundos para establecerse en la pantalla cuando se cambia el modo de video, o cuando la computadora y el monitor son encendidos. Algunos monitores toman más de 30 minutos en calentarse antes de producir una imagen estable. Si la imagen en el monitor está ligeramente inestable después de algún tiempo, probablemente sea porque el monitor no se encuentra sincronizando apropiadamente las señales que envía la tarjeta de video. El monitor necesita un ajuste vertical y horizontal, o ser reemplazado.

12) Un programa de juego en CGA distorsiona la imagen en un monitor analógico. El monitor puede no ser capaz de soportar ningún modo gráfico que sea requerido.

13) Utilizando una tarjeta aceleradora de video (monocromática o color) no se despliegan 44 renglones. El IBM ROM BIOS limita la pantalla a 25 renglones de caracteres.

14) No aparece nada en el monitor, pero tanto éste como la computadora parecen trabajar bien. Para verificar el estado del monitor se debe encender primero éste y después la computadora. Se deben checar los cables y la tarjeta de video. Verificar si el software está configurado correctamente.

II.8.2 TECLADO

El teclado es una parte importante del sistema de una computadora, ya que en muchas aplicaciones demasiado comunes es imposible realizar operación alguna sin un teclado.

No todos los teclados son iguales; algunos tienen las teclas ligeras y suaves, otros son de teclas duras. Algunos son ruidosos y otros silenciosos.

Los teclados tienen diferentes tipos de cubiertas para su protección, por ejemplo: cubiertas de plásticos especiales que protegen contra derrames, polvo o ambientes dañinos. En algunas áreas una cubierta es absolutamente esencial. Muchas de las cubiertas son fabricadas con plásticos suaves que son moldeados para cubrir las teclas.

Las máquinas de escribir son parcialmente estándar. Con solamente 26 letras del alfabeto y algunos símbolos, la mayor parte de las máquinas de escribir existentes tienen cerca 50 teclas. Las principales características de una máquina de escribir no han sido cambiadas en una computadora, pero algunas de las teclas importantes de control como Esc, Ctrl PtrScr, backslash y algunas otras, son cambiadas en la distribución del teclado. IBM decidió mover las teclas de funciones a la parte superior del teclado, arriba de las teclas numéricas.

Varias compañías han fabricado los teclados de 101 teclas del mismo tamaño que los de 84.

Cabe mencionar que todos los de PC-XT, AT, 80286, 80386 y 80486 tienen los mismos conectores. Cualquier teclado puede conectarse dentro de las máquinas mencionadas, pero los teclados de PC-XT tienen diferencias electrónicas y rastreo de frecuencias. Un viejo teclado XT puede ser conectado en una máquina 80286 o 80386, pero estos no pueden trabajar.

El teclado que se utiliza actualmente tiene asociada una computadora con un pequeño microprocesador y su propia ROM, la cual puede almacenar arriba de 20 o más tecleadas, y es posible determinar qué tecla fue presionada primero si se presionan dos a un mismo tiempo. Los nuevos microprocesadores para las computadoras del tipo AT son más complejos y sofisticados que en los primeros tipos de PC.

Actualmente muchos teclados tienen un pequeño switch en la parte trasera que les permite el uso del teclado XT al tipo AT 286, 386 o 486. Algunos de los nuevos teclados pueden, de manera electrónica, sensar el tipo de computadora y fijar automáticamente el switch adecuado.

* Tipos de teclado

Los teclados de 101 teclas fueron diseñados para reemplazar a los teclados de 83 y 84 teclas, y teóricamente reemplazar todos los teclados de cualquier sistema. Sin embargo, existía un pequeño problema para llevar a cabo el cambio de teclados: el sistema ROM BIOS de las computadoras podría no ser capaz de operar correctamente con los teclados de 101 teclas.

Se puede saber fácilmente cuando el sistema ha reconocido el teclado, ya que después de conectar el teclado a la computadora y encenderla, la luz del bloque numérico se mantiene encendida y mantiene habilitado al teclado numérico. Este método de detección no es 100% seguro, pero si el teclado mantiene encendida la luz del bloque de números, por lo regular el sistema soporta el teclado.

La función de detección no la realiza el teclado, sino la tarjeta madre (motherboard), para identificar el tipo de teclado que se tiene.

- Teclados de 83 teclas para PC-XT

Uno de los más criticados componentes de las originales PC-XT fue el teclado, esto debido al formato de 83 teclas. Las teclas Shift eran pequeñas y mal ubicadas en el lado izquierdo; la tecla Enter también era pequeña. Este tipo de errores de diseño se dieron gracias a que IBM producía las máquinas de escribir Selectric y quería mantener un estándar entre los formatos de los teclados.

- Teclados de 84 teclas para AT

Para el sistema AT también fue introducido el teclado de 84 teclas. Este nuevo teclado corrigió muchos de los problemas que tenía el anterior: la posición y arreglo del teclado numérico se modificó; la tecla Enter se diseñó más grande; las posiciones de las teclas Shift fueron corregidas en tamaño y colocación. Además, IBM añadió LEDs indicadores para conocer el estado de las teclas Caps Lock, Scroll Lock y Num Lock.

- Teclado de 101 teclas

IBM introdujó el teclado de 101 teclas ampliado para los modelos de computadora tipo XT y AT. Este nuevo teclado fue diseñado para conformar las regulaciones internacionales y especificaciones para teclados. Este tipo de teclados se fabricó en versiones con y sin los LEDs indicadores de estado, dependiendo del sistema que se vendiera (XT o AT).

El teclado de 101 teclas se divide en cuatro secciones:

- Area de teclado.
- Teclado numérico.
- Controles de cursor y pantalla.
- Teclas de funciones.

* Problemas con el Teclado

Todas las teclas en el teclado, cuando son presionadas o libreradas, transmiten y cortan códigos (*Scan Codes, códigos de rastreo*) al sistema de la computadora. El bit más significativo de cada código de rastreo es bajo cuando una tecla es presionada y alto cuando la tecla es liberada.

Cuando una tecla es presionada por un largo período, su código de rastreo puede ser transmitido contínuamente con una secuencia de retardo, en tanto se mantenga la tecla presionada. Si otra tecla es presionada, esto pueder causar que la primera detenga la repetición de su transmisión. El código de rastreo de la segunda tecla puede ser transmitido y la secuencia de retardo es reinicializada. El teclado ejecuta una autoprueba siempre que un arranque en frío o en caliente sea indicado por la computadora. La UCP del teclado verifica la localización de memoria de datos; la RAM interna y si alguna tecla se encuentra presionada o no. Si el diagnóstico de prueba es correcto el teclado transmite un código AAh a la computadora, este código es transmitido después de cada condición de encendido a menos que un error sea encontrado. Si la prueba de diagnóstico falla, el teclado transmite un código FDh/FCh a la computadora. Si una de las teclas es liberada mientras la computadora es reinicializada, se puede transmitir un código de error de rastreo a la computadora.

Los problemas con el teclado pueden ser provocados por las siguientes causas:

- Cuando la computadora es encendida los LED's del teclado no encienden: aparece un mensaje de error en la pantalla. Una vez que el sistema ha terminado su proceso de Inicio y el teclado no ha sido reconocido, el monitor no puede responder a ningún tecleo. Se debe checar el conector del teclado a la tarjeta madre o verificar si no han sido dañados los pines del mismo.

- Alguna tecla se encuentra pegada (la tecla es presionada y no regresa a su posición normal) debido al derrame de algún fluido sobre el teclado. Se debe desconectar el teclado del sistema, remover la tapa o cubierta de la tecla que está afectada y utilizar algún limpiador dentro del asta de la tecla. También puede utilizarse alcohol Isopropílico para verter una pequeña cantidad de éste en el switch del asta y limpiarla.

- Cuando un chip o un diodo se encuentra en mal estado dentro del teclado, puede causar intermitencia o ninguna respuesta por parte del teclado.

- Cuando se encuentra dañado un chip en la tarjeta madre, probablemente no se tenga respuesta del teclado, posiblemente puede aparecer un código de error 301 (ver apéndice C) en el monitor y la computadora no se puede inicializar. Se debe tratar de utilizar otro teclado para verificar si éste es el problema o lo es la tarjeta madre.

- Si alguna tecla es presionada y ésta no regresa a su posición, probablemente el resorte de contacto se ha roto. Si la tecla es un switch de contacto, cuando la tecla es presionada la resistencia del switch debe ser de 0 ohms y cuando es liberada deberá marcar infinito. En los teclados IBM PS/2 se tiene un pequeño resorte que es muy fácil de soltar, por lo que se debe tener cuidado al remover la tapa. Para un teclado clon, probablemente sea necesario desarmarlo, desoldar la tecla dañada y sustituirla por otra del mismo tipo.

- El buffer del teclado está lleno. Muchos teclados tienen un pequeño buffer de caracteres (16 generalmente), el cual esta provisto para prevenir pérdidas en el tecleo. Si se teclea demasiado rápido, el buffer del teclado puede tener un desbordamiento, y un caracter 00h es insertado dentro del buffer. El teclado transmitirá este código una vez a la parte superior del buffer que ha sido alcanzado. Cuando esto ha ocurrido, el teclado emitirá un "beep" cada vez que se teclea un caracter, para terminar con este problema, sólo se deja de teclear por algunos segundos para permitir que el buffer se limpie.

- Si durante el inicio del sistema se tiene presionada alguna tecla, un error de código de rastreo es desplegado en la pantalla y el sistema se detiene. Para terminar con el problema se debe realizar un arranque en caliente, presionando <Ctrl><Alt><Del> o un arranque en frío, evitando presionar alguna tecla durante el proceso.

II.8.3 MOUSE

Una de las grandes razones para el éxito de la Machintosh es que resulta fácil de utilizar. Con un mouse e iconos todo se puede realizar apuntando y presionando las teclas del mouse. Con el uso de éste y los iconos no es necesario aprenderse muchas instrucciones ni reglas.

El Mouse de Microsoft es estándar, por lo que muchas otras compañías emulan el sistema de esta compañía. Sin embargo, algunos tipos de mouse no se encuentran estandarizados; utilizan sistemas ópticos con un LED que brilla sobre una parrilla reflejante. Como el mouse se mueve a través de la parrilla, refleja la luz y ésta es tomada por un detector que envía a la computadora el movimiento del cursor.

Para un diseño que requiera tolerancias muy pequeñas de espaciamiento de la rejilla un mouse óptico puede no proveer la adecuada resolución. Esto se puede solucionar utilizando un mouse que contenga una bola en su interior.

El *ball mouse* o mouse de bola, contiene en su interior una "pelotita" de caucho que hace contacto con cualquier superficie plana. Cuando el mouse se mueve la bola gira. Dentro de él, dos pequeñas ruedas hacen contacto con la bola, una para el movimiento horizontal y otra para el movimiento vertical. La bola recoge suciedad, por lo que es necesario limpiarla frecuentemente.

El mouse de Machintosh tiene un solo botón, lo cual no da mucho a elegir, excepto por apuntar y presionar. Casi todos los mouse de PC tienen como mínimo dos botones, lo cual brinda al usuario tres elecciones: presionar el botón izquierdo, presionar el botón derecho o presionar ambos al mismo tiempo, según esté definido y según se requiera. Algunos mouse tienen tres botones de control, con ellos el usuario tiene siete opciones: botón izquierdo, botón medio, botón derecho, botones derecho y medio, botones izquierdo y medio, botones izquierdo y derecho y los tres botones al mismo tiempo. A pesar de todas estas opciones, la mayoría de los softwares existentes requieren utilizar sólo dos botones, uno a la vez.

El voltaje que requiere un mouse es regularmente de 5 volts. Algunos tienen un pequeño conector a un transformador que suministra la energía necesaria, otros permiten insertar un adaptador entre el cable conector del teclado y el conector de la tarjeta madre. Algunos mouse requieren el uso de uno de los puertos seriales de la computadora para tener entrada a ella. Pueden existir problemas si se tienen ocupados los puertos COM1 y COM2. Además, algunas

tarjetas madre tienen puertos construidos sobre ella misma pero no tienen espacio para usar una ranura (slot) e instalar una tarjeta para puerto.

Microsoft, Logitech y muchas otras compañías fabricantes de mouse han desarrollado un bus para mouse. La interfaz se conecta directamente con el bus y no requiere el uso de ningún puerto COM; sin embargo, el sistema requiere el uso de una de las ranuras de expansión de la computadora.

*** Trackballs**

Un trackball es un mouse con la parte inferior puesta hacia arriba. En analogía con el mouse común, un trackball puede tener el voltaje de alimentación de un transformador o de otro tipo de fuente. Este aparato requiere un puerto serial o un slot si es de tipo bus.

En este caso, en lugar de mover el mouse para que la bola se deslice, el trackball se mantiene fijo y se mueve la bola con los dedos. Usualmente los trackballs son tan grandes como la bola en un mouse, por lo que la posibilidad de obtener una mejor resolución es mayor.

Los trackball no necesitan tanto espacio de un escritorio como el mouse ordinario. El trackball MicroLINX tiene un conector que va en serie con el del teclado, lo que permite al sistema tomar sus necesidades de voltaje de la misma línea que la alimentación del teclado. El trackball ComLINX es idéntico al anterior con la excepción de que éste se conecta en un puerto serial.

II.8.4 PUERTO SERIAL/PARALELO

Un puerto le permite a una computadora controlar, utilizar y establecer comunicación entre otros equipos periféricos, tal como discos duros externos, impresoras, módems, mouse e incluso otras computadoras. Los puertos de comunicación más utilizados son los seriales a través de un conector *RS-232*, y los paralelos a través de un conector *Centronics*, estos generalmente se ubican en la parte posterior de la computadora.

Un puerto serial transmite o recibe datos bit a bit, en tanto que un paralelo los transmite o recibe byte a byte.

Existen otros tipos de puertos, tales como el *Bus Port*, puerto de ratón y puerto de juegos para conectar un joystick. El puerto SCSI se utiliza para conectar este tipo de periféricos.

Además de los puertos anteriores, también existe el puerto de comunicación de la señal de video, el cual varía de acuerdo al tipo de monitor.

Los puertos además tienen una distinción física: el número de pines y el tipo de conector. Es macho cuando los pines del puerto están hacia afuera, y es hembra si el puerto tiene los orificios correspondientes y el número exacto para aceptar los pines del macho.

Un puerto serial de la computadora generalmente es macho y el puerto de un módem es hembra. Para poder conectar ambos entre sí, se utiliza un cable RS-232 hembra a macho, conectando el lado macho de la computadora al hembra del cable y el lado macho del mismo al lado hembra del módem.

- Comunicación en línea del puerto serial

Para utilizar la comunicación en línea, se deben conocer los parámetros de envio y recepción del puerto (baud, velocidad, paridad, bits de datos y bits de alto). Los parámetros se cumplen mediante software especial de comunicación o utilizando el comando MODE de DOS, que ayuda también al direccionamiento adecuado de los puertos.

La transmisión asíncrona se realiza caracter a caracter y pueden pasar diferentes periodos de tiempo en el envío de ellos. No existen circuitos de reloj, por lo que los circuitos pueden procesar la información cuando sea recibida. Cada caracter lleva adjunta una señal de inicio y paro. Cuando se termina la recepción de la señal, el caracter es reconocido por dicha señal, y el tiempo que se toma para la recepción de los caracteres no importa. Este tipo de transmisión tiene 20% más utilización para el reconocimiento de cada caracter comparada con la transmisión síncrona. Los pines 2,3 y 7 son de los más importantes, y comúnmente los pines 1 a 8 y 20 son utilizados en comunicaciones seriales.

Con la transmisión síncrona los caracteres son transmitidos con una señal de reloj común que se aplica a los registros de recepción y transmisión. Las aplicaciones síncronas requieren que por lo menos los siguientes pines funcionen correctamente: 1, 2, 3, 4, 5, 6, 7, 8, 15, 17 y 20.

Algunas veces estos pines son conmutados por DTE y DCE pero la señal del pin 4 siempre fluye a través de DCE a DTE. Los conectores seriales hembras comúnmente son DCE, en tanto que los conectores seriales machos pueden ser ambos.

DCE (Data Communications Equipment, Equipo de Comunicación de Datos). Envía una señal al pin 4 cuando se quiere dar inicio al envío de datos, entonces los datos se envían al pin 2 y los datos son recibidos en el pin 3.

DTE (Data Terminal Equipment, Equipo Terminal de Datos). Se recibe la señal en el pin 4, envía los datos al pin 3 y recibe los datos en el pin 2.

El puerto serial tiene las siguientes características:

- 20mA de corriente en el lazo (loop) de la interfaz.
- 1 lógico = 20mA durante el pulso.
- 0 lógico = no hay flujo de corriente.
- Pin 18 +, recibe la corriente de datos del lazo.
- Pin 25 -, recibe la corriente de datos de regreso del lazo.
- Pin 9 +, transmite la corriente de datos de regreso.
- Pin 11 -, transmite corriente de datos al lazo.

Líneas de Datos. Un voltaje DC positivo mayor que +3V indica un 0 lógico, un valor de -3V indica un 1 lógico. Los voltajes pueden tener un rango de ±25V DC.

Líneas de Control. Las líneas de los niveles de señal de voltaje son las mismas que las de datos, excepto por los niveles de voltaje +3V o mayores indican CIERTO y niveles de voltaje de - 3V o más negativos indican FALSO.

Algunos problemas con adaptadores de puertos seriales pueden ser causados por:

1. Deficiencias de E/S de direcciones con otra tarjeta, los *jumpers* no se encuentran conectados correctamente. Para corregir este problema se debe hacer una revisión de las direcciones de las tarjetas que se utilizan, además de checar la configuración que se tenga con los jumpers. Se debe estar seguro que los puertos seriales (COM) estén definidos correctamente en su IRQ.

2. Mal funcionamiento de algún chip en la tarjeta adaptadora. Para llevar a cabo una adecuada revisión de este problema, se debe realizar la prueba de LOOP BACK, en la cual se utiliza un conector serial especial, y mediante un programa se realiza el envío de caracteres y se deben recibir los mismos, si no es así existe una falla.

3. La tarjeta de puertos no se encuentra en la ranura de expansión correctamente.

4. El software no se encuentra programado correctamente; por ejemplo, en el programa se despliegan COM3 y COM4, pero el equipo sólo reconoce COM1 y COM2.

5. Los cables utilizados no son los adecuados. Algunos cables de módem asíncronos no tiene los suficientes hilos para soportar las señales adicionales que se requieren para comunicaciones síncronas.

6. Se realizó mal soldadura en alguna unión. Se debe revisar la misma y si es posible corregirla.

7. No se encuentran instalados los circuitos para un segundo puerto serial.

8. Los circuitos del puerto no están conectados u orientados correctamente en su base.

9. La recepción o transmisión de caracteres se pierde; el puerto se encuentra cerrado. Se debe verificar que los chips de puerto serial utilizados para la computadora empleada son los adecuados, si no es así es necesario reemplazarlos por los correctos, que comúnmente se especifican en los manuales de cada computadora.

En la tabla 2.2, se proprocionan las velocidades de transmisión posibles en el puerto serial, así como las longitudes máximas de los cables a utilizar para ellas.

Velocidad (Bauds)	Máxima distancia (m)
300	1219.2
1200	304.8
1800	198.12
2400	152.4
4800	76.2
9600	45.72

Tabla 2.2 Velocidades de transmisión en el puerto serial y longitudes de cable permitidas

- Puerto Paralelo

El puerto paralelo está diseñado para conectar impresoras paralelas, pero también puede utilizarse como puerto general de E/S para cualquier dispositivo o aplicación que tenga las mismas capacidades de E/S.

El puerto de impresora típicamente tiene 12 puntos buffer TTL de salida. Estas salidas son tomadas y pueden ser escritas y leídas por medio de un programa de control, utilizando las instrucciones de procesador OUT o IN. El adaptador también tiene 5 estados permanentes de puntos de entrada, que pueden ser leídos utilizando la instrucción IN del procesador. Una entrada puede ser usada para crear una interrupción del procesador, dicha interrupción puede ser habilitada o deshabilitada por medio de un programa de control. Cuando el adaptador es utilizado para conectar una impresora paralela, los datos o comandos de la impresora son cargados en 8 bits, tomando el puerto de salida y la línea *Strobe* es activada, de éste modo se mandan datos a la impresora. El programa puede entonces leer la entrada del puerto de la impresora para conocer su estado, indicando en qué momento el siguiente caracter puede ser escrito, o si puede utilizar la línea de interrupción para indicar NOT BUSY (no ocupado) al software.

La señal de la línea de datos *Strobe* normalmente se mantiene alta; cuando la computadora quiere enviar datos a la impresora, la primera envía un pulso de salida bajo. La impresora carga los datos en el borde de subida del pulso.

Con la línea de reconocimiento, cada vez que la impresora recibe una señal de control de impresión o un byte de 8 bits de datos, se envía una señal de reconocimiento (pulso negativo) de regreso a la computadora.

La línea de ocupado (BUSY) manda una señal de regreso a la computadora informandole que el buffer se encuentra lleno y que no puede aceptar más datos en ese instante.

Los niveles de voltaje para señales paralelas son de 5V DC y se debe de colocar la impresora lo más cerca posible de la computadora. TIpicamente se utiliza un cable de 3.65m a 4.57m, si se utiliza un cable de mayor longitud es necesario que éste tenga una capacitancia muy baja para evitar perder cualquier señal. Para cables de mayor longitud se debe utilizar un convertidor paralelo a serial.

En general, uno de los problemas que ocurren con este tipo de puertos es que si se tiene instalada en el sistema una tarjeta adaptadora para impresora monocromática, ésta se reconoce como LPT1 y el puerto paralelo de una segunda tarjeta adaptadora es tomado automáticamente por defecto como LPT2. Sin embargo, a pesar de que los puertos se configuren con jumpers, el puerto LPT1 puede desconfigurarse y salir a LPT2. LPT3 no puede ser utilizado si se tiene una tarjeta CGA.

II.8.5 UNIDADES DE DISCO

* Cinta Magnética

La cinta magnética es un dispositivo de almacenamiento masivo, el cual continúa utilizándose debido a que es económica, compacta y funcional. Es compacta puesto que puede almacenar en una sola cinta 200 MB de información, y es funcional porque es fácil de instalar y guardar. El único defecto que pueden tener las cintas magnéticas, es que la información contenida sólo puede ser leída o grabada en serie y no en forma aleatoria como los discos duros o flexibles.

Las características principales de las cintas magnéticas son:

- Miden casi 0.5" de ancho, 1.5/1000 de pulgada de grosor y 300, 600, 1200 o 2400 pies de largo.
- Para realizar la grabación se utiliza una fina capa de óxido de fierro, dióxido de cromo o una mezcla de ambos, sostenida por una cinta plástica (comúnmente Mylar).
- La cinta puede ser plástica o metálica bañada en un material magnético.
- La información es almacenada como pequeños puntos magnetizados, ordenados en forma de columnas a través de la cinta.
- La lectura o grabación se realiza a una velocidad constante. Dicha velocidad varía de 75 a 200 pulgadas por segundo, leyendo o grabando desde 60 hasta 3000 KB por segundo.

El lector de cintas magnéticas es un aparato que contiene un controlador de cinta y cabezas de lectura y grabado. Este sistema es muy parecido al sistema giratorio de cassette para grabadora. La información representada por impulsos electrónicos similares a los de los discos duros y flexibles, es enviada o leída por la computadora a través de las cabezas de grabación/lectura de cintas magnéticas. Una cabeza de lectura y grabado se encuentra asociada con cada columna de puntos magnéticos, de tal forma que una columna pueda ser leída o grabada al instante mientras la cinta se mueva a través de las cabezas, éstas por lo general son fijas.

Los nombres utilizados con este tipo de unidades son: Tape Drive (IBM), Tape Controller (ICL), Tape Deck (NCR), Tape Unit (UNISYS).

En el caso de las PC, las unidades de cinta magnética se utilizan para realizar una copia de datos del disco duro, con el fin de conservar una fuente de respaldo en caso de suceder algún accidente.

* Floppys y Discos Duros

La memoria de disco magnético posee la característica de ser de acceso aleatorio. En una memoria de cinta, si nos encontramos en el extremo de la misma y necesitamos acceder al otro, debemos rebobinar la cinta, en tanto que en un disco, si nos encontramos en la parte más externa, podemos mover directamente la cabeza a la parte más interna sin detenernos en las pistas intermedias, esto trae como consecuencia, tener tiempos de acceso al disco mucho menores que en cinta. Los tiempos en disco se miden en décimas de milisegundos, en tanto que los de cinta son del orden de minutos.

Los discos giran a grandes velocidades y deben ser rígidos para mantener una pequeña zona de aire entre disco y cabeza.

En los sistemas de disco flexible, el disco y las cabezas hacen contacto mecánico de la misma forma que la cinta y la cabeza en el sistema de cinta. Además, el disco no debe ser totalmente rígido, puede ser delgado para ser ligeramente flexible. La velocidad de rotación del disco flexible es de 300 rpm, comparada con 3600 rpm para el disco rígido. El número de pistas es 77 comparadas con 500 o más del disco duro. La densidad de pistas es de 48 por pulgada comparada con 200 por pulgada. La capacidad de almacenamiento de un floppy es de aproximadamente 30 millones de bits por superficie, lo cual es superado por el disco duro en 10 veces para la misma superficie. Los tiempos de acceso están en el rango de cientos de milisegundos contra décimas de milisegundos del sistema de disco rígido.

Para manejarlos y protegerlos adecuadamente del ambiente, los discos se encuentran permanentemente encerrados en una envoltura de plástico.

- Anatomía Lógica de un Diskette

Los medios más usuales de almacenamiento masivo son el disco duro y el flexible. Este último (también llamado *floppy* o *diskette)*, tiene semejanza con un disco fonográfico, pero en lugar de surcos dispone de pistas y sectores magnéticos que permiten la organización lógica de la información. El diskette fue diseñado por el japonés Yoshiro Nakamats; el formato original fue de un disco de 8 pulgadas de diámetro.

Actualmente existen dos tipos de diskettes comerciales, compatibles con IBM:

1) Los de 5 1/4 pulgadas de diámetro, disponibles en versiones de 360 Kbytes y 1.2 Mbytes de capacidad.

2) Los de 3 1/2 pulgadas de diámetro, disponibles en versiones de 720 Kbytes y 1.44 Mbytes.

Recientemente los laboratorios de Sony Corp. están introduciendo al mercado discos flexibles de 2.88 Mbytes. También se han puesto a la venta diskettes de 3 1/2 pulgadas que trabajan con la ayuda de un rayo láser para la localización de los sectores, tecnología conocida como *floptical disk*, y con la cual se consiguen almacenar hasta 21 Mbytes.

- Estructura física de un diskette

El diskette es un dispositivo de memoria externa basada en la grabación/lectura de sectores y pistas sobre una superficie móvil magnetizable. Se encuentra fabricado con una lámina de plástico de mylar, recubierta de óxido magnético por ambos lados, y gira a 300 rpm.

En la figura 2.10 podemos observar las partes de un diskette de 3 1/2 pulgadas y en la figura 2.11 vemos las partes de un diskette de 5 1/4 pulgadas.

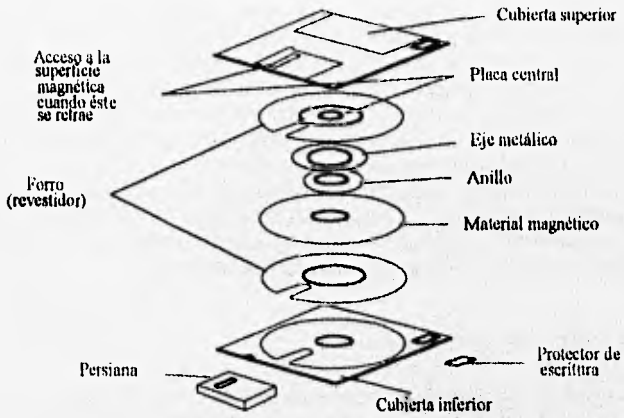


Figura 2.10 Partes de un Diskette de 3 1/2 pulgadas

La grabación y lectura de la información queda expresada en forma binaria y está a cargo de una cabeza de lectura/escritura constituida por un núcleo de hierro suave de alta permeabilidad, con una bobina enrollada y una pequeña separación, como se muestra en la figura 2.12.

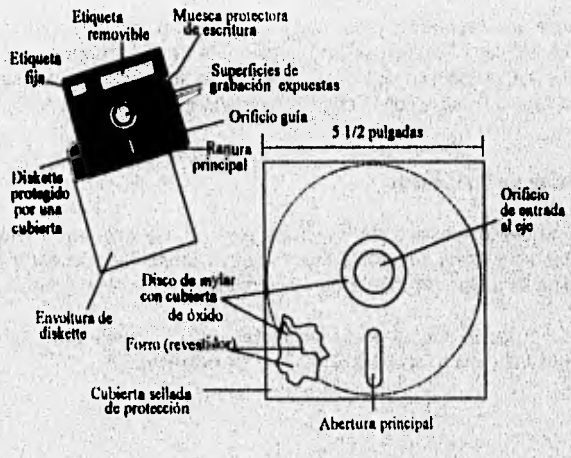


Figura 2.11 Partes de un Diskette de 5 1/4 pulgadas

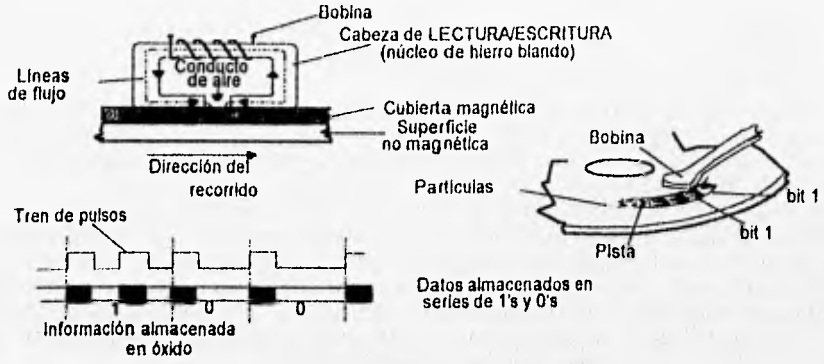


Figura 2.12 Lectura y Escritura de un Disco

Al escribir sobre la superficie del diskette fluye una corriente a través de la bobina, con lo que se forman líneas magnéticas de flujo que llegan al conducto de aire. Este tiene una baja reluctancia al flujo magnético que provoca que las líneas de flujo cambien de dirección y pasen a través de la cubierta magnética del diskette en rotación. Lo anterior origina canales de magnetismo que contienen la información en código binario.

La operación de lectura es inversa a la escritura, sólo que en este caso la bobina es el dispositivo sensor. Mientras el diskette gira, la superficie se mueve cerca de la cabeza de lectura/escritura y los canales que han sido magnetizados previamente producen un flujo a través del conducto de aire, y en el núcleo, induciendo una señal de voltaje que posteriormente se procesa y amplifica para las diversas actividades informáticas.

*** Formateo**

El formateo se divide en dos etapas:

- *Formateo físico o de bajo nivel*, que consiste en *dibujar* magnéticamente en cada cara del disco las pistas y sectores.

- *Formateo lógico o de alto nivel*, el cual es la continuación del anterior y consiste en dividir la capacidad de almacenamiento del disco en porciones que puedan ser manejadas por el sistema operativo.

- Formateo Físico

Durante el formateo físico se inicializa el diskette, para lo cual, un par de cabezas de lectura/escritura que se encuentran respectivamente con las dos superficies marcan magnéticamente cierta cantidad de pistas o tracks.

En un disco duro, se llama *cilíndro* al conjunto de pistas situadas perpendicularmente en la misma posición. En el caso del diskette solamente existen dos caras, por lo que cada cilíndro está formado por dos pistas, una colocada en el lado 0 y la otra en el lado 1. Lo anterior cambia de acuerdo a los diferentes discos duros.

Además de la formación del diskette en sectores y pistas, se necesita tener un sistema de direcciones que permita almacenar y accesar de manera ordenada y lógica la información, por lo cual es necesario numerar los cilíndros, caras, pistas y sectores. Existen dos modos de numeración: física y lógica.

En el caso de la numeración física, la tarjeta controladora del disco y las rutinas de la memoria ROM BIOS especifican el sistema de direccionamiento con base en 3 coordenadas: cara, pista y/o cilindro, y sector. Con la numeración lógica el sistema operativo considera los datos como si fueran una trenza de unidades secuenciales, por lo que numera los sectores de la pista que están situadas en la cara contraria, para continuar con el segundo sector de la cara. Con este método se minimizan los movimientos de las cabezas y se mejora el tiempo de acceso cuando se leen los sectores situados en forma ordenada.

Cabe mencionar que la unidad controladora de disco flexible reconoce la posición del primer sector de cada pista, mediante un pequeño orificio ubicado cerca del centro del diskette (tipo 5 1/4). El sistema operativo también reconoce la posición de los sectores por medio de este orificio. Para reconocer tal ubicación se proyecta un haz de luz sobre el orificio indicador, el cual únicamente pasa en el momento en que coincida con el orificio; cada vez que esto sucede durante la rotación, se puede ir determinando la posición del disco.

Pero además de los métodos de numeración anteriores, también se tiene un método de direccionamiento de la información llamado ***numeración de clusters***. El DOS agrupa un cierto número de sectores contiguos (*cluster*) y los maneja como un solo paquete de información en el disco.

- Formateo Lógico

Una vez creadas las pistas y sectores del disco, se debe utilizar un método que permita manejar con facilidad la información y llevar un conteo del estado actual de organización; tal método se define durante el formateo lógico.

En esta etapa, el DOS organiza la información en clusters como unidad mínima de almacenamiento. El número de sectores contenidos en cada cluster cambia dependiendo del tipo de diskette que se utilice.

El formateo lógico consiste en la creación de un conjunto de índices que indican en cualquier momento el estado del disco (zonas ocupadas, sectores defectuosos, espacio disponible y en qué lugar se encuentra), permitiendo direccionar la información del mismo. Para esto el disco se encuentra dividido en dos zonas:

1) Area del sistema:

a) Sector de arranque (Boot Sector)

Cuando se enciende la computadora, los programas de arranque de la ROM le ordenan que lea el sector de arranque del disco para comenzar a cargar en la RAM el sistema operativo. La computadora busca en primera instancia estas rutinas en el disco de la unidad A, pero si no las encuentra por estar vacía pasa enseguida a buscarlas al disco duro.

Si en la unidad A se introduce un disco auto-arrancable (que contiene los tres archivos básicos del sistema operativo) al encender la computadora, en RAM se cargarán las rutinas que le permitan trabajar, pero si el diskette no es auto-arrancable, se despliega un mensaje de error.

El sector de arranque debe estar formado de dos partes:

1) Un programa muy pequeño de arranque llamado IPL (*Initial Program Loading*).
2) Una lista de las características clave del disco (número de bytes por sector, número total de sectores por disco, cantidad de sectores por pista, número de cabezas de lectura/escritura, etc.).

Debido a la importancia de estos datos, todos los diskettes del DOS, sean o no auto-arrancables, deben disponer de un sector de arranque, que siempre se ubique, en el caso de este tipo de discos, en el primer sector de la primera pista de la primera cara: sector 1, pista 0, cara 0.

b) FAT (File Allocation Table)

La tabla de localización de archivos, *FAT*, es la segunda parte del sistema de disco y consiste en un área que registra tanto el estado general de todos los clusters como el direccionamiento de los archivos, según la cadena de clusters que ocupan.

En un diskette, un archivo rara vez queda grabado en un bloque de sectores contiguos, esto se debe a que el DOS distribuye los archivos por la superficie del disco aprovechando los espacios libres que van quedando por otros archivos que se han borrado o modificado.

En la FAT se asigna una entrada para cada cluster, que contiene la información respectiva de cada archivo, cuyo número de identificación corresponde con el número de dicho cluster.

c) Directorio raíz (Root Directory)

Este es un índice que el DOS crea en la última parte del área del sistema durante el formateo lógico. Tanto su tamaño como su posición quedan fijados durante esta etapa y no pueden alterarse posteriormente.

El número de sectores que ocupa el directorio raíz es distinto entre un diskette de baja densidad y uno de alta.

El directorio raíz dispone de un cierto número fijo de entradas, en las cuales se recoge la información esencial de cada archivo a almacenar en el diskette, con lo que la cantidad de archivos posibles de almacenar queda limitada por el número de entradas.

En la práctica, el directorio raíz se puede definir como el directorio de primer nivel en la estructura o árbol de archivos del disco, y se reconoce con la diagonal inversa (\), también llamada ***backslash***.

2) Zona de datos

Esta parte es el lugar donde se almacenan los archivos y programas del usuario. Es la continuación del área del sistema, prolongándose hasta el final del disco hacia el centro.

El DOS organiza la información en la zona de datos por clusters numerados secuencialmente, iniciando en todos los tipos de diskettes por el cluster 2; en consecuencia el último cluster tendrá asignado un número que es mayor en 1 al realmente existente, como espacio disponible para almacenamiento en el diskette.

*** Características Adicionales de los Discos Duros**

En general, los discos duros trabajan del mismo modo que las unidades de disco flexible, la principal diferencia se encuentra en la cantidad de datos que almacenan, puesto que se tratan de discos rígidos de aluminio.

Para conocer la capacidad total del disco duro lo único que debemos obtener es el producto resultante entre el número de cabezas, pistas o cilindros, sectores por pista y bytes por sector.

Además del conocimiento de la capacidad del disco, existen otros factores importantes para valorar el rendimiento de un disco duro:

- *Tiempo de posicionamiento (seek time).* Tiempo necesario para que el cabezal se ubique en la pista adecuada.
- *Tiempo de latencia (latency time).* Es el tiempo que el disco toma en espera de que el cabezal, una vez situado sobre la pista, acceda al sector adecuado.

- *Tiempo de transferencia (transfer time).* Es el tiempo que emplea el disco duro en enviar información hasta la memoria principal de la computadora.

- *Velocidad de transferencia (transfer rate).* Este factor se puede obtener dividiendo la cantidad de información transferida entre la suma del tiempo de posicionamiento, latencia y transferencia. Por lo regular, los fabricantes proporcionan los factores anteriores evitando así que el usuario realice las operaciones.

Otro tipo de conceptos utilizados cuando se define a un disco duro en el programa de SETUP y en el formateo del mismo son los siguientes:

- *Factor de Interleave.* Este factor consiste en la grabación de información en el mismo orden de sectores que se encuentra en el disco después de transferir la información. Con este tipo de sistema se aumenta la velocidad de acceso al disco. El factor se aplica debido a que normalmente el sector que se posiciona debajo de las cabezas no corresponde inmediatamente al sector leído a continuación, por lo que se debe esperar un tiempo a que el disco gire al siguiente sector.

- *Precompensación.* La velocidad lineal a la que giran los discos duros hace que las pistas del interior de los platos giren a menor velocidad que la de los exteriores, por lo que se debe compensar tal diferencia. Este factor se indica en el SETUP al especificar las características del disco, aplicándose un algoritmo de precompensación tanto en las operaciones de escritura como en las de lectura. Actualmente en la mayoría de los discos duros ya no es necesario especificar tal factor puesto que ellos solos son capaces de calcular y aplicar su propio algoritmo de precompensación.

- *Zona de aterrizaje*. Es la superficie en la que se posicionan las cabezas del disco cuando éste se detiene, en ella no existen ni se graban datos. Al girar los platos, estos provocan que su propia inercia hagan flotar las cabezas logrando hacer que no toquen la superficie del disco. Si el disco se detiene, las cabezas se pueden ubicar en una superficie no propia para ello. Si sucede que cuando las cabezas se encuentran apoyadas en dicha zona el disco llega a recibir un golpe, se podría dañar la superficie de éste, consecuentemente se perdería información y hasta podrían llegar a desaprenderse las cabezas. Cuando las cabezas se encuentran en la zona adecuada, los brazos se bloquean impidiéndose el desplazamiento.

Los factores anteriores son relativos para tener conocimiento de la efectividad del disco por lo que además de ellos, es conveniente conocer también el tipo de interfaz que se utiliza. Esta interfaz debe cumplir con todas las normas de estandarización.

- Interfaz del disco duro.

Actualmente existen en el mercado diversos modelos de disco duro disponibles, pero cuando se hace referencia a diferentes clases, se debe tomar en cuenta el tipo de interfaz o placa lógica que pueden tener.

La interfaz define el tipo de conexión física, las señales y el controlador, presentes en el sistema de disco. Los primeros modelos que se presentaban para las PC se encontraban dentro del grupo de los MFM (*Modified Frequency Modulation,* modulación por variación de frecuencia), con capacidades de 20, 40 y 80 Mbytes. Hoy en día este tipo de disco es muy inusual. Posteriormente aparecen los ESDI (*Enhanced Small Device Interface*, interfaz de dispositivo pequeño avanzado) para capacidades más grandes que los anteriores, pero actualmente también se encuentran obsoletos.

Las clases actuales de disco duro que se recomiendan y se encuentran disponibles son los IDE (*Integrated Drive Electronics*, electrónica de controlador incorporado), utilizados en computadoras de propósito general, y los SCSI (*Small Computer Systems Interface*, interfaz de pequeños sistemas de computadora) para un manejo más confiable y mayor volúmen de información.

La mayoría de las computadoras que se distribuyen actualmente tienen un sistema de disco tipo IDE, conocido también como *Bus AT*, en el cual no existe propiamente un controlador de disco ya que la misma interfaz realiza las funciones de éste, por lo que reduce a un adaptador que envía los datos de la interfaz del disco al bus de la computadora.

- Preparación del disco duro.

Al igual que en los floppys, el disco duro debe de recibir tanto el formateo de bajo como el de alto nivel. Una vez que el disco está formateado a bajo nivel se debe definir el área que se reserva para el sistema operativo, es decir, se tiene que particionar. Posteriormente se realiza el formateo de alto nivel, proporcionando los archivos necesarios para ejecutar el sistema operativo.

Al terminar de realizar lo anterior, el disco duro queda listo para que trabaje cuando se inicialice la computadora.

La planificación de discos se realiza de la siguiente manera:

- Los discos ópticos son capaces de almacenar una variedad de información digital como software de computadora o datos, información de audio digital y MIDI.

- Los sistemas ópticos, sin embargo, proveen grandes ventajas sobre su contraparte magnética. La densidad de almacenamiento de la media óptica es cerca de 30 tiempos mayor que la media magnética y el costo de almacenamiento es menor. El medio óptico es más durable y menos susceptible al calor, humedad, polvo, huellas digitales, campos magnéticos y vibraciones. El tiempo de vida del medio óptico es mucho mayor que los discos magnéticos y cintas.

**** El Disco Compacto.***

Los avances tecnológicos en optoelectrónica, procesamiento de señales digitales, corrección de errores, rastreador (scanner) óptico, modulación y tecnología láser ha sido incorporado dentro del sistema de disco compacto.

Un disco compacto contiene información codificada digitalmente en forma de orificios impresos dentro de la superficie. La información sobre el disco es leída por el lector óptico decodificando y procesando.

El diámetro del disco es de doce centímetros, con un grosor aproximado de 1.2 mm. El orificio central tiene 15 mm de diámetro y permite que el disco pueda ser colocado en el reproductor de CD.

La velocidad angular del disco se decrementa conforme el lector óptico se mueve en dirección exterior a la pista (track). A una velocidad lineal de 1.2 m/s, la velocidad angular varía entre 486 y 196 rpm. A 1.4 m/s la velocidad angular varía entre 568 y 228 rpm. Estos orificios varían en longitud desde 0.833 hasta 3.054 nm dependiendo de la codificación de datos y la velocidad lineal del disco. La información contenida en la estructura del orificio se interpreta como un uno lógico siempre que el orificio exista, y se considera un cero lógico a todos los espacios existentes entre ellos. El ancho y profundidad de los orificios son aproximadamente de 0.5 y 0.11 nm respectivamente. Los orificios son localizados en un track en forma de espiral con un grado de inclinación de 1.6 nm; el número total de espirales contenidas en el track del disco es de 20 825.

Las especificaciones del disco compacto fueron conjuntamente desarrolladas por Philips Corporation y Sony Corporation, y son documentadas en un manual conocido como el "*Libro Rojo*" (Red Book).

- Estructura

El disco compacto medio consta de un substrato transparente de policarbonato cubierto por un material reflectivo, mismo que está cubierto por una capa protectora. El rótulo de los orificios se encuentra sobre esta capa. El substrato permite que el rayo láser penetre en la capa reflectiva. El tamaño del punto del rayo láser se encuentra enfocado inicialmente con un diámetro de 0.8 nm a la superficie del disco, para 1.7 nm a la superficie reflectiva.

La longitud de onda del láser en el aire es de 780 nm ; sobre el substrato tiene un índice de refracción de 1.55, la longitud de onda es reducida aproximadamente a 500 nm. Las profundidades de los orificios son de 110 y 130 nm, y están diseñados para ser aproximadamente de 1/4 de la longitud de onda del láser. La profundidad del orificio de 1/4, crea una estructura de difracción tal que la luz reflejada sufre interferencia destructiva antes del primer rayo reflejado. Esta interferencia hace que decremente la intensidad de luz reflejada

que recibe el lente. La presencia de orificios y polvo son detectadas por los sensores ópticos como cambios de intensidad de luz. Las señales de luz son convertidas a su correspondiente señal eléctrica por los fotodetectores.

*** CD-ROM**

CD ROM, este nombre se encuentra formado por las siglas en inglés que *significan "disco compacto de solo lectura"*. El CD ROM es un disco de plástico similar a los discos compactos de audio, diseñados para tener grabada la información de cualquier tipo y ser recuperada en una computadora.

A raíz del gran éxito logrado por la industria de los discos compactos de audio, con tirajes de miles o millones de copias, los costos de producción bajaron y permitieron que fuera rentable iniciar la producción de CD ROM, los cuales son producidos en las mismas plantas donde elaboran los de audio.

La información grabada solo puede ser leída: no hay posibilidad de borrarla, modificarla o añadir datos, debido a que la grabación de la información es física y no magnética. La lectura se hace con un lector óptico especial que se añade a una computadora.

El CD ROM tiene varias características que han permitido un enorme crecimiento de esta industria, la primera es su capacidad de almacenamiento que puede llegar hasta 660 MB, es decir, se puede almacenar el equivalente a unas 350 000 cuartillas. En segundo lugar, el costo de almacenamiento y reproducción es muy bajo; por ejemplo, el costo de almacenamiento masivo de una base de datos en CD ROM es de 0.02 dólares por MB. En tercer lugar, el formato del CD ROM (ISO 9660) se encuentra estandarizado, esto significa que cualquier disco puede ser leído en cualquier lector y aún cuando ésta condición no se cumple totalmente, más del 95% de los discos sigue dicho formato.

El concepto fundamental del CD ROM es que se puede disponer de un enorme volumen de información en un pequeño disco de plástico, dependiendo únicamente de la computadora personal, sin necesidad de estar ligado a una red o a un sistema en línea. El CD ROM permite tener una base de datos reproducida *n* veces, tantas como sea necesario, a un bajo costo.

La industria del CD ROM se inicia a mediados de la decáda de los 80's. Las primeras aplicaciones estuvieron orientadas a editar discos con información de referencias bibliográficas y hemerográficas especializadas, que rápidamente se fueron ampliando en otras áreas informativas.

Los costos para elaborar un CD ROM pueden ser muy variables. El mayor costo está asociado con la construcción de la base, es decir, el llevar la información a un medio electrónico y depurarla; en caso de que la información esté en papel, los costos de captura pueden ser muy elevados.

*** Problemas con los Drives y con los Discos**

Los problemas con unidades de disco pueden ser causados por:

1) Cabeza de lectura y escritura sucia.
2) Diskette defectuoso; los archivos se encuentran demasiado fragmentados.

3) Diskette de diferente densidad insertado en la unidad; el diskette fue formateado con un drive con mal alineamiento y diferente velocidad.
4) Fluctuaciones de potencia del toma corriente de AC o de la fuente de poder interna.
5) El adaptador de la unidad de disco se encuentra defectuoso o no está instalado correctamente.
6) El cable del adaptador hacia el drive no está conectado correctamente; el cable tiene un hilo que se encuentra roto o retorcido; los contactos en el adaptador y en la unidad de disco están sucios.
7) La unidad de disco está desalineada.
8) La velocidad del motor no es la correcta.
9) Problemas con el software. Se encuentra colocado el protector contra escritura.
10) El DOS se encuentra dañado.
11) La unidad de disco está dañada.
12) Algún papel u otro objeto extraño se encuentra dentro de la unidad.
13) La fuente y/u otros dispositivos no se encuentran instalados adecuadamente.
14) Diskette insertado por alguno de los lados, parte posterior o superior; la unidad seleccionada no es la correcta.

Los problemas con el disco duro pueden ser causados por:

1) Media defectuosa en el sector de arranque. Se debe reformatear el disco a bajo nivel.
2) El motor se encuentra en mal estado. Checar las conexiones del mismo, asegurarse de la conexión de la fuente y verificar la soldadura del conector de esta.
3) Existencia de circuitos del disco en mal estado. Es necesario verificar que los chips se encuentren orientados adecuadamente y bien colocados en su socket; checar todas las conexiones; reemplazar la tarjeta lógica.
4) La tarjeta controladora se encuentra en mal estado. Reinstalar la tarjeta en el socket de bus de expansión; reinstalar todos los chips en su base; verificar el conjunto de jumpers; verificar si es posible soportar el tipo de disco duro; verificar cables y conexiones.
5) El disco duro y el controlador no son del mismo tipo (ESDI, RLL, MFM, SCSI). Se debe de reemplazar el controlador o el disco.
6) El disco no se encuentra formateado correctamente. Se debe de realizar un formateo a bajo nivel, identificando todas las pistas en mal estado.
7) Los cables de la tarjeta controladora al disco duro están dañados.
8) Los cables están defectuosos, reemplazarlos y checar si los pines de la tarjeta controladora se encuentran o no doblados.
9) Los cables no se encuentran conectados en los pines correspondientes. El pin 1 generalmente se estará identificando con la señal "↓", moldeada en el conector de plástico, también puede ser identificado como el hilo en rojo sobre el cable.
10) Bajo o nulo nivel de potencia en el disco duro. Verificar el conector de la fuente así como ésta. La tarjeta lógica del disco puede estar fallando.
11) Los conectores están flojos en el disco. Se deben volver a colocar, y los pines deben coincidir con las entradas de los conectores.
12) La tabla de configuración en el SETUP de CMOS de la tarjeta madre es incorrecto. Verificar que el BIOS de la tarjeta madre pueda soportar el número de cabezas y cilindros de ese disco duro. Si se esta utilizando un controlador de XT en un sistema AT, se debe de comunicar a BIOS que el disco duro instalado para este controlador no esta instalado.

13) Tabla de partición incorrecta. La partición primaria puede solamente tener un máximo de 32 Mb.
14) El manejador puede estar en conflicto con otro software; se tendrá que modificar el archivo CONFIG.SYS y remover el manejador de disco duro, entonces se podrá ejecutar el software problema desde el disco duro.
15) La velocidad del bus es demasiado rápida para la tarjeta controladora del disco duro. Disminuir la velocidad del bus si es posible, esto se puede realizar por medio de un jumper o con el BIOS de la tarjeta madre. Tratar de insertar la tarjeta en otra ranura.
16) Existen muy pocos estados de espera para la operación del controlador. Los datos del disco duro pueden mezclarse. Para resolver el problema hay que seguir el paso anterior.
17) Se tiene un inusual número de sectores con error. La ejecución de formato a bajo nivel identifica pistas en mal estado; se debe verificar su correcta ejecución, así como la fuente de poder, tarjeta controladora, cables y cualquier manejador del disco duro. Examinar si existen problemas con los estados de espera y la velocidad del bus. El software utilizado también puede ser el problema.
18) Pequeños ruidos son emitidos por el disco duro. El ensamble de la cabeza de lectura/escritura puede estar roto o los platos pueden estar fuera de balance. Si el disco se encuentra montado en una posición vertical, tratar de colocarlo horizontalmente. Si el ensamble de la cabeza lectura/escritura está rota, es probable que se necesite cambiar el disco.
19) Si la rutina de arranque no se ejecuta adecuadamente, el sector de arranque posiblemente se encuentra dañado o la geometría del disco duro no es compatible con el BIOS de la computadora

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CAPITULO III : IMPRESORAS Y PROTOCOLOS

III.1 TIPOS DE IMPRESORAS

Después del monitor, la impresora es el periférico que más se utiliza dentro de los sistemas computacionales (al desplegar información). La impresora tiene como función obtener copias impresas de los programas y/o datos que trabaja la computadora.

La impresora como periférico, debe estar bajo las ordenes de la computadora, por lo que la comunicación entre ambas debe ser lo más clara posible, a esta comunicación se le llama *protocolo.*

Cuando se conecta la impresora a la computadora, ésta envía una señal indicando que se encuentra lista para recibir la información; se dice que la impresora está configurada.

Así como se han venido desarrollando sistemas en software y hardware, de igual modo han existido avances dentro del mundo de los periféricos utilizados. Las impresoras también se encuentran con diferentes características, las que se explican en forma genérica a continuación:

*** Impresión por impacto**

Para la impresión por impacto se utilizan dos técnicas: *impresión por caracteres completos* y el *sistema de formación de caracteres.*

La *impresión por caracteres completos* es similar a la técnica utilizada en las máquinas de escribir mecánicas, por lo que da un resultado similar a la de una máquina de alta calidad. La mayoría de las impresoras que realizan el trabajo mencionado son las llamadas de tipo margarita. Se les llama de este modo debido a la disposición que tiene el cabezal en el cual se encuentran los caracteres. Se trata de un disco de metal o de plástico duro al final del cual se encuentra el molde de cada caracter. La impresora hace girar el disco hasta que se posiciona en el caracter deseado, después un martillo golpea éste contra la cinta y el papel produciéndose así la impresión.

En la *técnica de formación de caracteres* (también llamada *técnica de matriz de puntos*), los caracteres son el resultado de la impresión de una serie de puntos dispuestos de una forma determinada. A partir de una matriz de puntos pueden generarse todos los caracteras necesarios, de acuerdo a la disposición de la impresora.

Algunas impresoras del tipo de matriz de puntos pueden modificar el tamaño y estilo de los caracteres.

La impresión matricial se lleva a cabo con un cabezal de agujas; sin embargo, el sistema es muy ruidoso debido al movimiento de éstas.

* Impresión de Chorro de Tinta

Para este tipo de impresión, se inyecta un chorro de tinta sobre el papel. No se requiere de algún tipo de papel en especial, pero se recomienda que éste reciba cierto tratamiento antes de ser utilizado.

Existen dos métodos para lanzar tinta desde el surtidor: el contínuo y el de descarga a demanda. En el primero, la tinta se descarga de manera constante y es desviada por un campo eléctrico hacia donde se requiera. En la descarga a demanda sólo se lanza tinta cuando se requiere.

*** Impresión de Transferencia Térmica**

Esta técnica se lleva a cabo haciendo que la cinta de impresión deposite en el papel la forma del caracter deseado, gracias a la acción de un comando de orígen térmico generado por el cabezal de impresión.

En este tipo de impresoras se tiene la ventaja de eliminar completamente el ruido que se presenta en las impresoras por impacto, su calidad es mayor y también su velocidad de impresión. Sin embargo, su precio es considerablemente más alto que el de las mencionadas anteriormente y por lo mismo son poco utilizadas.

*** Impresión Electrostática**

Esta técnica la utilizan las impresoras láser. Consiste en un tambor cargado eléctricamente que atrae al *toner* (depósito de tinta) para pasarlo posteriormente al papel.

El principal inconveniente en la utilización de este método de impresión, es la necesidad de utilizar papel especial que contiene un material que retiene la carga eléctrica.

La impresión electrostática opera por la aplicación de carga eléctrica que es recogida por un electrodo ubicado detrás del papel. Cuando ocurre lo anterior, es activado un pixel específico del papel que acepta el toner.

III.1.1 IMPRESORAS DE MATRIZ DE PUNTOS

Las impresoras de este tipo tienen como característica importante el poseer un cabezal de forma de matriz de puntos, el cual tiene movimiento con el fin de formar al caracter deseado y, una vez que esto sucede, realizar el golpe sobre la cinta para imprimirlo en el papel.

Las impresoras de matriz de puntos cuentan con las siguientes partes fundamentales:

- Cable de conexión eléctrica.
- Cable de conexión a la UCP.
- Cinta de impresión.
- Guías de papel (posterior y laterales).
- Unidad de arrastre para papel continuo.

Los elementos mencionados varían de una impresora a otra de acuerdo al modelo y marca a la que se refiera. La figura 3.1 muestra algunas de las partes mencionadas.

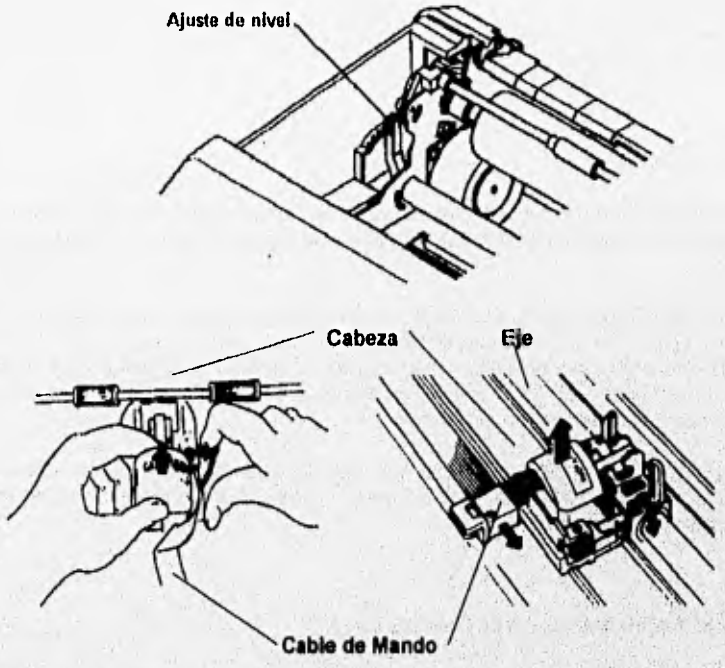


Figura 3.1 Algunas Partes de la Impresora de Matriz de Puntos

La impresora puede separarse en dos partes para su explicación:

- *Parte externa*, que se compone fundamentalmente por el interruptor de alimentación; una cubierta delantera, cuya función es proteger al sistema interior del medio ambiente; una palanca de selección, que sirve para escoger el tipo de papel (continuo u hojas sueltas) que se va a utilizar; la unidad de arrastre, que es necesaria cuando se utiliza papel continuo; el panel de control, que se compone por una serie de teclas e indicadores de estado de la impresora. Algunas impresoras poseen un espacio para introducir una tarjeta que contiene diferentes tipos de letras. También se pueden apreciar en la parte externa de la impresora los cables de conexión, tanto para la alimentación como para la comunicación con la computadora.
- Parte interna, en ella se encuentran los circuitos de alimentación, control y memoria de la impresora fundamentalmente.

*** Tipos de cabezales**

El cabezal de las impresoras de matriz de puntos está formado por una serie de agujas, por ello, el número de caracteres diferentes que pueden ser impresos es mayor mientras el número de agujas utilizadas para formarlo también lo sea. La clasificación del tipo de cabezal dependerá del número de agujas con que se cuente; las impresoras matriciales comunes se encuentran con cabezales de 9, 24 y 48 agujas.

*** Tipos de cintas**

La cinta es parte importante de la eficiencia de la impresora. Las cintas pueden clasificarse de acuerdo a sus formas, pero cada una es diferente según el tipo de impresora que se esté utilizando.

De acuerdo a los colores que componen las cintas, éstas pueden ser *monocromas* (negras o azules) y *policromas*. Estas últimas tienen una distribución de franjas horizontales de colores (generalmente azúl, rojo y amarillo) que en ocasiones también contienen el color negro. Con este tipo de cintas es posible la impresión en diferentes colores al combinarlos entre ellos para obtener colores intermedios o cualquier otro.

Por último, es importante mencionar que algunas cintas también pueden usarse como correctoras, ya que contienen una sustancia química que cubre el caracter impreso, funcionando así como borrador.

III.1.2 IMPRESORAS DE INYECCION DE TINTA

Estas impresoras tienen como principal diferencia respecto a las demás, un mecanismo llamado de chorro de tinta. Este mecanismo se compone de un cartucho en el cuál se encuentra la tinta líquida, y es expulsada a través de impulsos que son sometidos al cartucho. Dichos impulsos serán unos más fuertes que otros, de acuerdo a la calidad de impresión que se quiera dar.

Las impresores de inyección de tinta pueden también imprimir a colores y en este tipo de impresiones puede notarse una mayor calidad, sobre todo en la impresión de imágenes. De acuerdo a una serie de colores fundamentales se obtiene una gran variedad de tonos.

Físicamente, la impresora de inyección de tinta cuenta con las siguientes partes fundamentales:

- Cebador: Restaura el flujo de tinta expulsando las burbujas de aire.
- Limpiacartuchos: Elimina la tinta y las partículas de las placas de la boquilla del cartucho de impresión.
- Carro: Contiene el cartucho de impresión que expulsa la tinta a lo largo de su recorrido.
- Barra: Mantiene el papel presionado contra el rodillo en el momento de la impresión.

- Cepillo: Elimina la tinta seca de los conectores metálicos que se encuentran en el carro.
- Conectores Metálicos: se encuentran en contacto con el papel por un extremo, y en contacto con el cartucho de tinta en el otro; posicionan la tinta sobre la línea que barre el carro.
- Conmutadores: Se encuentran en la parte posterior y se utilizan para seleccionar el tamaño del papel.
- Conector para interfaz: Permite la comunicación entre la computadora y la impresora a través del cable correspondiente.
- Alimentación: Es la entrada de voltaje con que opera la impresora.

Las impresoras de inyección de tinta trabajan con cartuchos de tinta que son colocados de forma específica de acuerdo al tipo de éste y al modelo de la impresora.

Generalmente es recomendable cebar y limpiar el cartucho antes de instalarlo (Ver Figura 3.2).

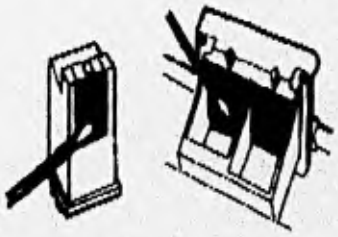


Figura 3.2a Limpieza del Cartucho de Tinta

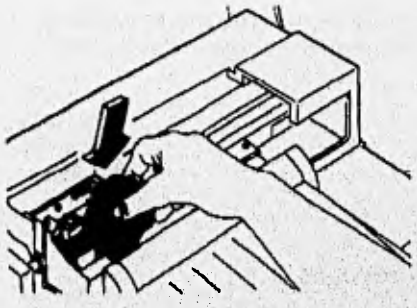


Figura 3.2b Colocación del Cartucho de Tinta

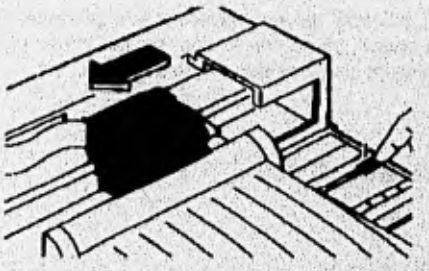


Figura 3.2c Movimiento del Cartucho de Tinta

Figura 3.2 Limpieza e Instalación de cartucho en impresoras de inyección de tinta

III.1.3 IMPRESORAS LASER

Las impresoras láser tienen como principio fundamental la utilización de un tambor cargado electrostáticamente, el cuál atrae al toner para poder realizar la impresión en el papel.

El trabajo que llevan a cabo este tipo de impresoras se produce en diferentes pasos: el primero consiste en la recepción de los datos que son enviados por la computadora, de los cuales la impresora hace una "imagen modelo" que posteriormente se almacena en la memoria de la misma. Una vez que los datos se encuentran en la memoria, se procede a vaciar la información en el tambor, cuando éste gira su superficie recibe una carga eléctrica uniforme a través de una corona de descarga desde el cable cargador principal que gira paralelo a lo largo del tambor. Posteriormente, un rayo láser se refleja en un espejo poligonal rotacional y recorre todo el tambor, con ello se obtiene una imagen, que posteriormente es revelada aplicando el toner al tambor. El toner se lleva al tambor por medio de un rodillo magnético, y se adhiere a las áreas eléctricamente neutras para ser impresas en negro.

Al término de los procesos anteriores se transfiere la imagen al papel.

Al igual que con los otros tipos de impresoras, en las láser es importante la configuración de la misma, con las siguientes recomendaciones de selección fundamentales:

- Emulación
- Tamaño de papel
- Formato de página
- Interfaz

III.2 PROTOCOLOS

Los protocolos son una serie de instrucciones programables que permiten configurar las impresoras de manera que el usuario pueda obtener características específicas de impresión, por ejemplo, ajuste de márgenes, calidad de impresión, espacios, entre otras.

A continuación se incluyen algunos protocolos importantes correspondientes a las impresoras EPSON de agujas, ya que es un tipo "estándar" que puede ser compatible en la mayoría de las impresoras de matriz de puntos (de 9 agujas).

*** Protocolos para impresoras EPSON**

- Control Vertical

El control vertical maneja las distancias de avance o retroceso del papel; para seleccionar diferentes distancias se tienen las siguientes instrucciones:

Avance de hoja

Distancia	Instrucción	Decimal	Hexadecimal	Observaciones
1/8"	ESC 0	27 48	1B 30	Avanza o retrocede
7/72"	ESC 1	27 49	1B 31	Avanza o retrocede
1/6"	ESC 2	27 50	1B 32	Avanza o retrocede
n/216"	ESC 3 n	27 51 n	1B 33 n	Avanza o retrocede
n/72"	ESC A n	27 65 n	1B 41 n	Avanza o retrocede

Avance de Línea LF (Line Feed)

Distancia	Instrucción	Decimal	Hexadecimal	Observaciones
	9	10	0A	Impresión de línea actual y avance a la siguiente
n/216"	ESC J n	27 74 n	1B 4A n	Valor de n entre 0 y 255
n/216"	ESC j n	27 106 n	1B 6A n	Avance en sentido inverso. n entre 0 y 255
n	ESC f1 n	27 102 1n	1B 66 1n	Desplazar n líneas a partir de la actual. n entre 0 y 127
n	ESC C n	27 67 n	1B 43 n	Selección de longitud de página de n líneas con el espacio actual configurado. n entre 1 y 127
n	ESC C 0 n	27 67 0n	1B 43 0n	Selección de longitud de página en n cm. con espaciado actual configurado. n entre 1 y 127

Avance de página. FF (Form feed).

Instrucción	Decimal	Hexadecimal	Observaciones
'	12	0C	Avance al principio de la siguiente página y desplaza la posición de impresión al margen izquierdo
ESC B n1... nx0	27 66 n1 ... nx0	1B 42 n1...nx0	Tabulación vertical. Sustitución de nuevos valores de paradas de tabulación (hasta 16), fijadas por el valor de n.
ESC e 1n	27 101 1n	1B 65 1n	Tabulación vertical cada n líneas. Cancelación de las paradas existentes fijandose las nuevas de acuerdo al valor de n (entre 1 y 127)
ESC / n	27 47 n	1B 2F n	Selección del canal de tabulación vertical en función del valor de n (entre 0 y 7)
ESC b m n1... nx0	27 98 m n1... nx0	1B 62 m n1 ... nx0	Se establecen paradas de tabulación vertical en canal. Cancelación de las paradas de tabulación en el canal n (entre 0 y 7), estableciendo nuevas paradas para ese canal

Tabulación vertical. VT (Vertical Tab).

Instrucción	Decimal	Hexadecimal	Observaciones
0	11	0B	El papel avanza hasta la siguiente parada de tabulación vertical, desplazándose hasta el márgen izquierdo. Si se encuentra en la última parada, avanza hasta el principio de la siguiente página

- Mandatos de Control Horizontal

Instrucción	Decimal	Hexadecimal	Observaciones
ESC l n	27 108 n	1B 6C n	El márgen izquierdo es seleccionado en la columna n (entre 0 y 299), según el espacio actual
ESC Q n	27 81 n	1B 51 n	El márgen derecho es seleccionado en la columna n (entre 1 y 127), con el actual espaciado de caracteres
CR	13	0D	Retorno de carro
BS	08	08	Backspace
ESC a 0	27 97 0	1B 61 0	Justificación a la izquierda
ESC a 2	27 97 2	1B 61 2	Justificación a la derecha
ESC a 1	27 97 1	1B 61 1	Centrar texto
ESC D n1...nx0	27 68 n1 ... nx0	1B 44 n1 ... nx0	Fijación de las paradas de tabulación horizontal hasta un máximo de 32. n entre 1 y 255. Las paradas deben especificarse en modo ascendente
ESC e 0 n	27 101 0 n	1B 65 0 n	Selección de paradas de tabulación horizontal cada n columnas. n entre 0 y 255. Cualquier alteración de estos límites mostrará un error de lectura
HT	09	09	Tabulación horizontal. Se desplaza la posición actual de la cabeza de impresión a la siguiente parada de tabulación que tenga establecida
ESC n1 n2	27 92 n1 n2	1B 5C n1 n2	Desplaza la impresión a derecha o izquierda, un máximo de 6". n1 en pulgadas, n2 = posición del cabezal
ESC # n1 n2	27 36 n1 n2	1D 24 n1 n2	Tabulación horizontal absoluta en pulgadas, con un máximo de 8"
ESC f 0 n	27 102 0 n	1B 66 0 n	Desplazamiento del cursor n espacios, entre 0 y 127

- Mandatos de Control de Espaciado y Tamaño de los Caracteres

Instrucción	Decimal	Hexadecimal	Observaciones
ESC P	27 80	1B 50	Espaciado pica (Pica Pitch)
ESC M	27 77	1B 4D	Espaciado elite (Elit Pitch)
ESC SI	27 15	1B 0F	Impresión condensada. Se cambia el espaciado a pica condensado
DC2	18	12	Cancelar impresión condensada

Continua...			
Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC W 1	27 87 1	1B 57 1	Impresión expandida. Los caracteres se imprimen en doble anchura
ESC W 0	27 87 0	1B 57 0	Cancelar impresión expandida
ESC S 0	27 14	1B 0E	Impresión expandida por una sola línea
DC4	20	14	Cancelación de impresión expandida por una sola línea
ESC p 0	27 112 0	1B 70 0	Selección de espacio fijo
ESC p 1	27 112 1	1B 70 1	Selección de espacio proporcional
ESC ! n	27 52 n	1B 21 n	Selección maestra de impresión, de acuerdo al valor de n y a las características que se deseen obtener: grande - 32 repicado - 16 enfatizada - 8 condensada - 4 elite - 1 pica - 0 El valor de n se obtiene al sumar las combinaciones deseadas

- Control de Fuentes

Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC 4	27 53	1B 34	Seleccionar caracteres en cursiva o itálica
ESC 5	27 54	1B 35	Cancelar impresión en cursiva o itálica
ESC E	27 69	1B 45	Seleccionar impresión en negrita. No puede ser usado con superíndices o subíndices
ESC F	27 70	1B 46	Cancelar impresión en negrita
ESC G	27 71	1B 47	Seleccionar impresión en repicado o doble. No puede ser usado con superíndices o subíndices
ESC H	27 72	1B 48	Cancelar impresión en repicado o doble
ESC -1	27 45 1	1B 2D 1	Seleccionar subrayado
ESC -0	27 45 0	1B 2D 0	Cancelar subrayado
ESC S 0	27 83 0	1B 53 1	Seleccionar superíndices
ESC T	27 84	1B 54	Cancelar superíndices o subíndices

- Mandatos de Control de Gráficos

Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC K n1 n2	27 75 n1 n2	1B 4B n1 n2	Impresión de gráficos de 8 bits de densidad simple
ESC L n1 n2	27 76 n1 n2	1B 4C n1 n2	Impresión de gráficos de 8 bits de densidad doble
ESC Y n1 n2	27 89 n1 n2	1B 59 n1 n2	Impresión de gráficos de 8 bits de doble densidad en alta velocidad

Continua...			
Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC Z n1 n2	27 90 n1 n2	1B 5A n1 n2	Impresión de gráficos de 8 bits de densidad cuádruple
ESC ? n m	27 63 n m	1B 2A n m	Conversión de la densidad de gráficos
ESC * m n1 n2	27 42 m n1 n2	1B 2A m n1 n2	Selección del modo de gráficos

- ***Mandatos de Selección de Juegos de Caracteres***

Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC 7	27 55	1B 37	Seleccionar juego de caracteres # 1 de IBM
ESC 6	27 54	1B 36	Seleccionar juego de caracteres # 2 de IBM

- ***Mandatos Varios***

Instrucción	**Decimal**	**Hexadecimal**	**Observaciones**
ESC Space	27 62	1B 3E	Fijar el bit más significativo (MSB) en uno
ESC =	27 61	1B 3D	Fijar el MSB en cero
ESC #	27 35	1B 23	Aceptar el MSB tal y como es
DEL	127	7F	Borrar el último caracter mandado
CAN	24	18	Cancelar
DC3	19	13	Colocar la impresora en estado Off-line
DC1	17	11	Colocar la impresora en modo On-line
BEL	07	07	Campana
ESC @	27 64	1B 40	Reinicializar la impresora

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CAPITULO IV : ANALISIS DE RENDIMIENTO

IV.1 MEDICIONES DEL RENDIMIENTO

Con la palabra "*rendimiento*" se desea dar a conocer la eficiencia con la que un sistema de computación cumple sus objetivos, por lo que el rendimiento se toma como *una "cantidad relativa más que absoluta"*, aunque en algunas ocasiones se mencionan "*medidas absolutas de rendimiento*" como el número de trabajos por hora que un sistema computacional puede servir.

Algunas medidas del rendimiento, así como los tiempos de respuesta, se dice que están orientadas al usuario, otras, como la utilización de la UCP, se dice que están orientadas al sistema. Las medidas del rendimiento más comunes son:

- *Tiempo de regreso.* Por ejemplo, en un sistema de procesamiento por lotes, éste se conoce como el tiempo de entrega del trabajo hasta el regreso al usuario.

- *Tiempo de respuesta.* Comúnmente se le conoce como el tiempo que transcurre desde que el usuario presiona la tecla ENTER hasta que el sistema despliega una respuesta.

- *Tiempo de reacción del sistema.* Es el tiempo que transcurre desde que el usuario presiona la tecla ENTER hasta que se da tiempo de servicio o respuesta a la petición del usuario.

Las mediciones de tiempo anteriores dan lugar a otras medidas del rendimiento utilizadas frecuentemente que son:

- *Varianza de los tiempos de respuesta.* Es una medida de dispersión. Una varianza pequeña indica que los tiempos de respuesta experimentados por los usuarios se encuentran muy próximos a la media. Con una varianza grande tenemos indicación de que algunos usuarios pueden experimentar tiempos diferentes a la media. La varianza de tiempos de respuesta nos indica, principalmente, si los usuarios están teniéndo un servicio rápido o muy lento.

- *Capacidad de ejecución.* Es la medida de ejecución de trabajo por unidad de tiempo.

- *Carga de trabajo.* Es la medida de la cantidad de trabajo que se ha introducido al sistema y que éste debe de procesar con un buen desempeño.

- *Capacidad.* Medida del rendimiento máximo que un sistema debe tener, siempre que éste se encuentre listo para recibir más trabajos y exista alguno disponible inmediatamente.

- *Utilización.* Es el tiempo en el que un recurso se encuentra en uso. La utilización puede llegar a ser una medida ineficiente, esto debido a que los procesos a ejecutar no siempre se encuentran en ciclos infinitos para lograr una alta utilización de la UCP, cuando esto se consigue, la UCP debe presentarse en cualquier instante como:

i) disponible

ii) en estado de programa
iii) en estado de supervisión

IV.2 FACTORES IMPORTANTES QUE AFECTAN EL RENDIMIENTO

Aproximadamente hace 25 años, la mayor parte de los costos relativos a un equipo de cómputo se basaban en el hardware. Con los avances tecnológicos los costos han disminuido notablemente. Pero como contraparte, los costos de trabajo han aumentado paulatinamente absorbiendo gran parte del costo computacional, esto da como consecuencia el rediseño del rendimiento del hardware base y la medición adecuada del rendimiento a la productividad humana.

Con la utilización del microprocesador a partir de la década de los 70's ha sido posible proporcionar la UCP a un costo adecuado. La evaluación del rendimiento de un equipo computacional debe realizarse principalmente en los dispositivos de entrada/salida, ya que en ellos los costos se mantienen altos.

Para la evaluación del rendimiento se mencionan tres objetivos:

- *Evaluación de selección.* El evaluador de rendimiento debe decidir si la adquisición de un sistema de computación de un proveedor en particular es adecuada.

- *Proyección del rendimiento.* En este caso se debe estimar el rendimiento de un sistema ficticio, el cual puede ser un sistema nuevo o un componente se software o hardware nuevo.

- *Control del rendimiento.* El evaluador debe de reunir datos sobre el rendimiento de un sistema o componente en existencia con el fin de asegurar que estos cumplan con las metas de rendimiento requeridas. Es importante conocer lo anterior, puesto que es una ayuda para tomar decisiones en cuanto a la conveniencia de modificar un sistema de prioridades de trabajo existente.

La evaluación y predicción del rendimiento son necesarias desde la creación de un nuevo sistema, así como la verificación de operación diaria del mismo después de su instalación, para llevar a cabo las consideraciones de modificación o posible reemplazo por un mejor sistema. El proveedor trata de predecir en las primeras etapas de desarrollo de un nuevo sistema lo siguiente:

- La naturaleza de las aplicaciones que se utilizaran en el sistema.
- Las cargas de trabajo anticipadas a las aplicaciones que se deberán manejar.

Después de que el proveedor da inicio al desarrollo e implementación de un nuevo sistema, la evaluación y predicción del rendimiento se utilizan para determinar la mejor organización del hardware; las estrategias de administración de recursos que se deberán implementar en el sistema operativo, y si el sistema evolutivo cumple o no con los objetivos de rendimiento.

Al ser lanzado el producto al mercado, el proveedor debe estar preparado para responder diferentes preguntas del usuario, como si el sistema puede manejar aplicaciones con ciertos niveles de rendimiento. Los usuarios suelen estar interesados en la elección de una configuración adecuada de un sistema que pueda servir a sus necesidades. Esta adecuación muchas veces es llamada "*proceso de configuración*".

La *sintonización del sistema*, (que consiste en la obtención de un rendimiento óptimo), a menudo, puede provocar mejoras notables en el rendimiento, una vez que el sistema está ajustado a las características de la instalación del usuario.

IV.3 TECNICAS DE EVALUACION DEL RENDIMIENTO

En la siguiente tabla se muestran las técnicas y aplicaciones a diferentes propósitos del rendimiento. Cada término se evalúa por separado para considerarlos adecuadamente según los rendimientos del hardware y del software.

Técnica de evaluación	Propósito de evaluación					
	Evaluación de selección (el sistema existe en algún otro lugar)		Proyección del rendimiento (el sistema aún no existe)		Control del rendimiento (el sistema está en operación)	
	Hardware nuevo	Software nuevo	Diseño de hardware nuevo	Diseño de software nuevo	Nueva forma del hardware	Cambiar software
Tiempos	1	-	1	-	-	-
Mezclas	1	-	1	-	-	-
Núcleo	2	1	2	1	-	-
Modelos	2	1	2	1	2	-
Puntos de referencia	3	3	-	2	2	2
Programas sintéticos	3	3	2	2	2	2
Simulación	3	3	3	3	3	3
Monitor (hardware y software)	2	2	2	2	3	3

Técnica no aplicable:

1) Ha sido usada pero es inadecuada.
2) Proporciona alguna ayuda, pero es insuficiente, se debe utilizar en combinación con otras técnicas.
3) Satisfactoria.

Tabla 4.1 Técnicas de Evaluación del Rendimiento

Los tiempos proporcionan las herramientas para realizar rápidas comparaciones del hardware en una computadora.

*** Mezcla de instrucciones**

Esta técnica utiliza un promedio de varios tiempos de las instrucciones más adecuadas en una aplicación determinada.

El evaluador de rendimiento analiza la mezcla de trabajos de una instalación predeterminada y trata de obtener un promedio de los tiempos de las instrucciones utilizadas con más frecuencia en el sistema. En ese momento, las computadoras pueden ser comparadas con mayor eficacia que la que proporcionan los tiempos por si solos. Además, las mezclas previenen al evaluador para utilizar otras técnicas antes de tomar alguna decisión de adquisición de algún equipo de cómputo. Las mezclas proporcionan poca o ninguna información útil para evaluar al software.

*** Programas del núcleo**

Los tiempos y las mezclas solo realizan énfasis a algunos aspectos de un conjunto de instrucciones del sistema computacional. Un *programa núcleo* es un programa común que se puede ejecutar en un sistema.

Utilizando los tiempos proporcionados por el fabricante, se ocupa un programa núcleo que se cronometra en una máquina dada, entonces se realizan las comparaciones entre las diferentes máquinas y los tiempos de ejecución del programa núcleo.

Los núcleos proporcionan mejores resultados que los tiempos y las mezclas de operaciones, pero con ellos se realiza un mayor trabajo manual de preparación y tiempo. La ventaja de los núcleos es que estos son programas completos, y será lo que el usuario ejecutará en el sistema computacional que se tome en cuenta.

*** Modelos analíticos**

Son representaciones matemáticas o componentes de sistemas de computación. Este tipo de modelos tienen ciertas desventajas, por ejemplo, los evaluadores deben de ser expertos en matemáticas y casi nunca tienen el conocimiento adecuado del procesamiento de datos en un ambiente comercial. Otro inconveniente que se presenta es cuando el problema es muy complejo. En este caso, el evaluador tiene menor posibilidad de encontrar una solución adecuada que describa el comportamiento del modelo.

*** Puntos de referencia**

Un punto de referencia es un programa de comparación del rendimiento que el evaluador ejecuta en el sistema que se está caracterizando. Regularmente, un punto de referencia es

considerado como un programa que se ejecuta con frecuencia en una instalación. El evaluador se familiariza con el rendimiento del punto de referencia del equipo en existencia, y cuando éste se ejecuta en un nuevo equipo, el evaluador puede llegar a tener conclusiones adecuadas del mismo.

Los puntos de referencia ya existen, por lo que el evaluador solo elige entre los programas comerciales conocidos. En este caso, la posibilidad de un error humano es mínima, comparado con los tiempos, mezclas o núcleos, puesto que la computadora ejecuta el punto de referencia, mientras que el tiempo se mide con cronómetros o con otros medios.

Los puntos de referencia son utilizados en la evaluación del hardware y del software, además son muy utilizados en la comparación de sistemas antes y después de realizar algún tipo de cambio en ellos. Sin embargo, no son útiles en la predicción de efectos en cambios propuestos, a menos de que exista otro sistema con los cambios incorporados en el que los puntos de referencia se pueden ejecutar.

*** Programas sintéticos**

Este tipo de programas llevan a cabo la combinación de las técnicas de los núcleos y los puntos de referencia; son de tipo de real y se han diseñado para dar a conocer características específicas de una computadora. Una de las grandes ventajas que se tienen sobre los puntos de referencia es que un programa sintético puede aplicarse para probar una característica particular de una nueva máquina que puede o no existir.

*** Simulación**

Es una técnica en la cual el evaluador desarrolla un modelo por computadora del sistema que se está evaluando. El modelo se prueba en un sistema computacional que pueda realizar millones de operaciones detalladas con rapidez y precisión, de esta forma se conoce el comportamiento del sistema mediante un tiempo de simulación.

Los simuladores regularmente se dividen en:

- *Manejados por los eventos.* Se controlan por eventos producidos en el simulador de acuerdo a distribuciones de probabilidad.
- *Manejados por libreto.* Se controlan por datos que se obtienen empíricamente y son manejados cuidadosamente para obtener el comportamiento anticipado del sistema en simulación.

*** Control de rendimiento**

Este tipo de control es una recolección y análisis de información relativa al rendimiento de sistemas existentes; ayuda a la determinación del rendimiento de un sistema relacionado con la capacidad de ejecución, tiempos de respuesta, predecibilidad, etc. El control de rendimiento localiza embotellamientos rápidamente y ayuda a administrar el mejoramiento del rendimiento.

Para localizar embotellamientos se utilizan los rastreos de ejecución de instrucciones o rastreos de ejecución de módulos.

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CAPITULO V: DISEÑO E IMPLEMENTACION DE INTERFACES

V.1 VERIFICACION DE EQUIPO

El diagnóstico del equipo de cómputo, implica una revisión de las características básicas del equipo: número de puertos, número de unidades de disco, coprocesador matemático, memoria base, identificador de equipo (modelo, submodelo, revisión), fecha de BIOS, nombre del fabricante del BIOS, tipo de tarjeta de video, nombre del fabricante de la tarjeta de video, etc.

Para realizar este procedimiento se utilizan programas de diagnóstico, tales como: Norton Utilities (Sysinfo), CheckIt, PCTools, etc. Pero en ocasiones se carece de estos programas teniendo que esperar a conseguirlos, o en su defecto, buscar las especificaciones de la computadora y las tarjetas instaladas, resultando esto un proceso tedioso. La alternativa más viable en esta situación, es el uso del programa *DEBUG* incluido dentro de las utilerías de **MS-DOS**.

El programa debug normalmente se utiliza para la inspección y prueba de programas, en nuestro caso se utilizará como un medio de acceso a las áreas de ROM BIOS.

La tabla de contenido de ROM BIOS se encuentra a partir de la dirección 0040:0000, describiendo las características del equipo.

Direcciones de puertos seriales:

0040h:0000h	Dirección del primer puerto serial RS-232	COM1: Word
0040h:0002h	Dirección del segundo puerto serial RS-232	COM2: Word
0040h:0004h	Dirección del tercer puerto serial RS-232	COM3: Word
0040h:0006h	Dirección del cuarto puerto serial RS-232	COM4: Word

Direcciones de puertos paralelos:

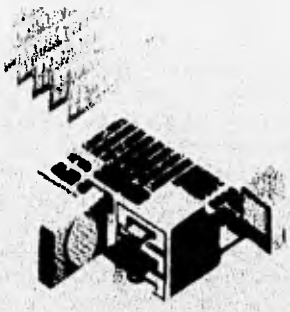
0040h:0008h	Dirección del primer puerto paralelo	LPT1: Word
0040h:000Ah	Dirección del segundo puerto paralelo	LPT2: Word
0040h:000Ch	Dirección del tercer puerto paralelo	LPT3: Word
0040h:000Eh	Dirección del cuarto puerto paralelo	LPT4: Word

Equipo:

0040h:0010h	Descripción del equipo	Word

La descripción del equipo consiste de dos bytes, que proporcionan información acerca de algunos elementos que integran la configuración de la computadora: disco duro, tarjeta para mouse, coprocesador matemático, número de unidades de disco, número de puertos seriales instalados, número de adaptadores para impresora instalados, etc.

Bit 0	DiskBoot (Bit 0 = 1, presente)
Bit 1	Coprocesador Matemático (Bit 1 = 1, presente)
Bit 2	Tarjeta para mouse (Bit 2 = 1, presente)
Bit 3	No analizado (Reservado)
Bit 4, 5	No analizados (modo inicial de video)
Bit 6, 7	Número de Drives instalados (Agregar al valor 1)
Bit 8	No analizado (DMA)
Bit 9, 10, 11	Número de puertos COM
Bit 12	GAME/IO (Bit 12 = 1, presente)
Bit 13	No analizado (Attached Serial Printer)
Bit 14, 15	Número de adaptadores de impresora instalados



Memoria base:

0040h:0013h	Cantidad de memoria base instalada	Word

Banderas del teclado:

0040h:0017h	Banderas de teclado Insert, NumLock, CapsLock, ScrollLock, Alt, Ctrl	Byte

Bit 7	= 1 Insert activo
Bit 6	= 1 CapsLock activo
Bit 5	= 1 NumLock activo
Bit 4	= 1 ScrollLock activo
Bit 3	= 1 Alt presionado
Bit 2	= 1 Ctrl presionado
Bit 1	= 1 Left Shift presionado
Bit 0	= 1 Right Shift presionado

Modo de video:

0040h:0049h	Modo de video	Byte

Número de columnas desplegables:

0040h:004Ah	Número de columnas	Word

Tamaño de la página de video

0040h:004Ch	Tamaño de página de video	Word

Contador de tiempo:

0040h:006Ch	Contador de tiempo	4Bytes

El contador de tiempo registra el número de pulsos emitidos a partir de la media noche, al instante en que es consultada esta dirección. El reloj interno pulsa 18.2065 veces por segundo.

Discos fijos:

0040h:0075h	Número de unidades de disco fijo instaladas	Byte

Número de renglones desplegables:

0040h:0084h	Número de renglones desplegables	Byte

Ancho del caracter:

0040h:0085h	Ancho del caracter	Byte

Bits de control de video VGA:

0040h:0087h	Bits de control de video VGA	Byte

La dirección 0040h:0087 proporciona información del tipo de dispositivo de video instalado en el equipo. La inspección de esta dirección se realiza considerando lo siguiente:

Bit 0 — 0 Trasladar el cursor a los modos 0-3 usando Monitor ECD en 350 líneas.
1 Se inhibe la traslación del cursor

Bit 1 — 0 Monitor a color ó ECD conectado al adaptador de video.
1 Monitor monocromático conectado al adaptador de video.

Bit 2 — 1 Espera por monitor
Bit 3 — 0 Adaptador EGA/VGA Activo.
Bit 4 — X No utilizado

Bit 6, 5 — Tipo de Memoria instalada en el adaptador

$(00)_2$ = 64K
$(01)_2$ = 128K
$(10)_2$ = 192K
$(11)_2$ = 256K

Bit 7 — 0 Clear RAM Disable
1 Clear RAM Enable

Switch de datos EGA/VGA

0040h:0088h	Switch de datos EGA/VGA	Byte

Bits 7, 6, 5, 4 — Bits característicos del conector 3-0, respectivamente
Bits 3, 2, 1, 0 — Switch de opciones 3-0, respectivamente

Bits de control EGA/VGA

0040h:0089h	Switch de datos EGA/VGA	Byte

Bit 7	= 1	200 líneas
Bit 6, 5		Reservado
Bit 4	= 1	400 líneas
Bit 3	= 1	Paleta no cargada
Bit 2	= 1	Monitor monocromático
Bit 1	= 1	Escala de grises
Bit 0	= 1	Reservado

Información de velocidad de datos en disco flexible:

0040h:008Bh	Información de velocidad de datos en diskette	Byte

Bits 7, 6 Ultima velocidad de datos fijada por el controlador:

$(00)_2$ = 500 Kbits/s
$(01)_2$ = 300 Kbits/s
$(10)_2$ = 250 Kbits/s

Bits 5, 4 Ultimo paso seleccionado para el disco
Bits 3, 2 Velocidad de transferencia en la operación inicial:

$(00)_2$ = 500 Kbits/s
$(01)_2$ = 300 Kbits/s
$(10)_2$ = 250 Kbits/s

Bits 1, 0 Reservados

Información del controlador de disco flexible

0040h:008Fh	Información del controlador de diskette	Byte

Bit 7	Reservado
Bit 6 = 1	Un drive determinado para drive 1
Bit 5 = 1	Drive 1 es multivelocidad
Bit 4 = 1	Drive 1 soporta cambio de línea
Bit 3	Reservado
Bit 2 = 1	Un drive determinado para drive 0
Bit 1 = 1	Drive 0 es multivelocidad
Bit 0 = 1	Drive 0 soporta cambio de línea

Tipo de media en ambos drives:

0040h:0090h-91h	Tipo de media en ambos drives	Byte

La dirección 0040h:0090h corresponde al drive 0, y la 0040h:0091 al drive 1.

Bit 7, 6 Velocidad de transferencia:

$(00)_2$ = 500 Kbits/s
$(01)_2$ = 300 Kbits/s
$(10)_2$ = 250 Kbits/s

Bit 5 = 1 Requiere doble paso (360K media en 1.2 Mb Drive)
Bit 4 = 1 Media reconocida
Bit 3 Reservado
Bit 2, 1, 0 Definición para el usuario:

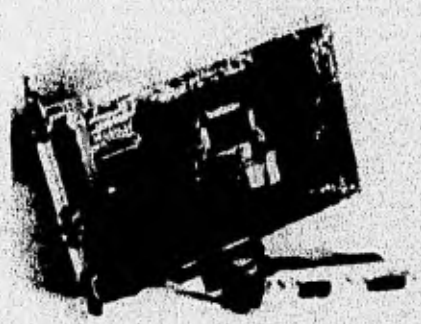
$(111)_2$ = 720K media en 720K ó 1.44Mb drive;
ó 1.44Mb media en 1.44Mb Drive
$(101)_2$ = Conocida, 1.2Mb media en 1.2Mb Drive
$(100)_2$ = Conocida, 360Kb media en 1.2Mb Drive
$(011)_2$ = Conocida, 360Kb media en 360Kb Drive
$(010)_2$ = Molesta*, 1.2Mb media en 1.2Mb Drive
$(001)_2$ = Molesta*, 360Kb media en 1.2Mb Drive
$(000)_2$ = Molesta*, 360Kb media en 360Kb Drive

* **Nota:** No corresponde a la capacidad real del drive, la capacidad lógica no corresponde a la física o a la definida por el drive.

*** Obteniendo direcciones de los puertos.**

Para realizar esta operación, cargamos el programa DEBUG:

DEBUG 



A continuación aparecerá un guión como indicador o prompt del programa, este nos indica que DEBUG está listo para recibir instrucciones. Para desplegar el contenido en la dirección deseada se da el comando D (Dump):

```
-D 0040:0000

0040:0000  F8 03 F8 02 00 00 00 00-78 03 78 02 00 00 00 00   ........x.x.....
0040:0010  63 84 00 7F 02 80 00 20-00 00 3C 00 3C 00 78 2D   c.......  ..<.<.x-
0040:0020  74 14 0D 1C 44 20 20 39-30 0B 30 0B 34 05 30 0B   t...D  90.0.4.0.
0040:0030  3A 27 30 0B 30 0B 30 0B-30 0B 0D 1C 74 14 02 10   :'0.0.0.0...t...
0040:0040  21 00 01 00 00 00 00 06-02 03 50 00 00 10 00 00   !.........P.....
0040:0050  00 10 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:0060  0F 0E 00 D4 03 29 30 03-00 00 C0 FF D5 4F 0C 00   .....)0......O..
0040:0070  00 00 00 00 00 01 00 00-14 14 14 14 01 01 01 01   ................
```

Normalmente el comando desplegará 128 bytes, pero se puede limitar la salida agregando el comando L (Load):

```
-D 0040:0000 L 10

0040:0000  F8 03 F8 02 00 00 00 00-78 03 78 02 00 00 00 00   ........x.x.....
```

En este ejemplo, se desea sólo desplegar los primeros 16 bytes, por lo cual se agrega el valor 10h adelante del comando L.

El obtener la dirección de los puertos seriales o paralelos, implica leer el contenido en la dirección asignada para cada uno de ellos, recordando que la dirección se encuentra almacenada en dos bytes, formando un valor de tipo word (palabra). Por ejemplo, para obtener la dirección del primer puerto serial, se debe consultar la dirección 0040h:0000h, leyendo dos bytes:

```
-D 0040:0000 L 02

0040:0000  F8 03                                            ........x.x.....
```

El primer byte desplegado es el menos significativo, mientras que el segundo representa el más significativo. De esta manera, la dirección correspondiente al primer puerto serial es 03F8h. Cuando el valor almacenado es igual a 0000, el puerto no se encuentra disponible debido a una configuración inadecuada del mismo o ausencia de la tarjeta correspondiente.

Este procedimiento es válido para cada uno de los puertos, por ejemplo el primer puerto paralelo:

```
-D 0040:0008 L 02

0040:0000                  78 03                            ........x.x.....
```

Para este caso, la dirección queda definida como 0378.

-D 0040:0000 L 10

0040:0000 F8 03 F8 02 00 00 00 00-78 03 78 02 00 00 00 00 ....x.x.....

Puerto	Dirección	Puerto	Dirección
COM1	03F8	LPT1	0378
COM2	02F8	LPT2	0278
COM4	0000	LPT3	0000
COM4	0000	LPT4	0000

Nota: El valor de 0000h indica la ausencia del puerto correspondiente

*** Descripción de elementos del equipo.**

Los elementos que componen una computadora, tales como: disco de arranque, coprocesador matemático, tarjeta para mouse, unidades de disco, puertos (seriales, paralelos); son descritos en la dirección 0040h:0010h de la tabla ROM BIOS, consistiendo en dos bytes. Cada bit representa a algún elemento de la PC:

Bit 0	DiskBoot (Bit 0 = 1, presente)
Bit 1	Coprocesador Matemático (Bit 1 = 1, presente)
Bit 2	Tarjeta para mouse (Bit 2 = 1, presente)
Bit 3	No analizado (Reservado)
Bit 4, 5	No analizados (modo inicial de video)
Bit 6, 7	Número de Drives instalados (Agregar al valor 1)
Bit 8	No analizado (DMA)
Bit 9, 10, 11	Número de puertos COM
Bit 12	GAME/IO (Bit 12 = 1, presente)
Bit 13	No analizado (Attached Serial Printer)
Bit 14, 15	Número de adaptadores de impresora instalados

Usando el programa debug y su comando D (Dump), accesamos a la dirección antes mencionada:

- D 0040:0010 L 02

0040:0010 63 84 c..... ..<.<.x-

Considerando que el primer byte desplegado representa el menos significativo y el segundo el más significativo, tenemos el valor de $(8463)_{16} = (1000010001100011)_2$; vaciando este dato en la siguiente tabla:

Disco de arranque	Presente	Número de unidades de disco flexible	2
Coprocesador Matemático	Presente	Número de Puertos Seriales	2
Tarjeta para mouse	No Presente	Número de adaptadores de impresora	2
Puerto para Juegos	No Presente		

Para conocer el número de unidades de discos fijos se consulta el byte almacenado en la dirección 0040h:0075h, de la tabla de BIOS:

-D 0040:0075 L 1

0040:0070 01 .....

En este ejemplo, el número de discos fijos instalados es uno.

*** Memoria base instalada.**

Para calcular la cantidad de memoria base instalada en el equipo, se tiene que hacer referencia nuevamente a la tabla de descripción de ROM BIOS, en este caso la dirección asignada es la 0040h:0013h (recordando que el valor contenido en esta dirección consta de dos bytes).

La lectura de la cantidad de memoria se realiza tomando en cuenta que el primer byte es el menos significativo y el segundo el más significativo. Además, el valor contenido representa el número de bloques de 1024 bytes cada uno. Para desplegar este valor, se utiliza nuevamente el programa debug y su comando D (Dump) en la dirección antes señalada.

```
-D 0040:0013 L 2

0040:0010                80 02                                    c..... ..<.<.x-
```

El valor desplegado es el 0280h que corresponde al valor de 840 bloques de 1024 bytes, lo que nos da un total de 655360 bytes disponibles de memoria base, que normalmente son los reportados por algunos programas ó utilerías del sistema operativo:

Memoria_Base = (NumBloques X 1024) Bytes

Memoria_Base = (640 X 1024) Bytes
Memoria_Base = 655360 Bytes

*** Verificación de Banderas del teclado.**

La verificación del teclado incluye entre otras cosas, verificar el buen estado de las teclas y el estado que guardan los LED's indicadores de las banderas CapsLock, NumLock y ScrollLock.

Las funciones comunes de lectura de caracteres, desde el teclado, no son capaces de reconocer la pulsación de las teclas Alt, Ctrl, Left Shift, Right Shift, Insert, CapsLock, NumLock y ScrollLock. Para realizar el rastreo, de estas teclas, es necesario modificar el valor contenido en la dirección 0040h:0017h; el cual se encuentra contenido en un byte. El byte contenido en esta dirección nos proporciona información acerca del estado de las teclas en cuestión, correspondiendo a cada una de ellas un bit de estado.

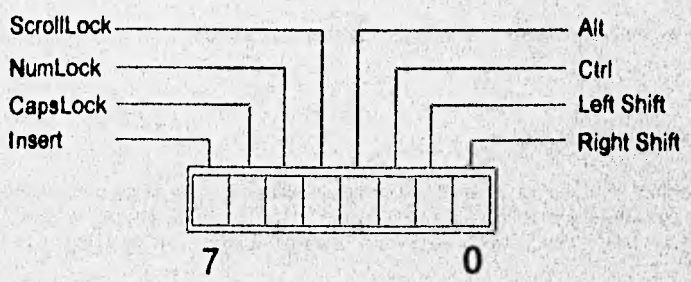


Figura 5.1 Banderas del Teclado

Dentro de este conjunto de bits, los primeros cuatro son utilizados para conocer el estado que guardan las teclas especiales que representan (ver Fig. 5.1); los otros cuatro son utilizados para modificar y conocer el estado que guardan sus correspondientes teclas.

- Acceso y lectura de valores de estado.

Para accesar al byte de estado del teclado, es necesario apuntar hacia esa dirección por medio de una variable absoluta o una variable de tipo apuntador. La primera resulta ser la más adecuada, ya que el acceso es directo; sin embargo, este tipo de variables no pueden ser direccionadas hacia otra localidad de memoria. La implementación de este método en Turbo Pascal, se muestra a continuación:

```
Uses CRT;

Var
      KeyBrd      : Byte Absolute $0040:$0017;
      ApuntaKB    : ^Byte;

Begin
  ApuntaKB := Ptr($0040, $0017);
  WriteLn('ApuntaKB := ', ApuntaKB^:3);
  WriteLn('KeyBrd :=', KeyBrd:3);
End.
```

Suponiendo que sólo la función de CapsLock se encuentra activa, el programa desplegará los siguientes valores:

```
ApuntaKB  := 64;
  KeyBrd  := 64;
```

En este caso el bit 6 se encuentra en estado alto ó 1, representando $(64)_{10} = (01000000)_2$. Ahora, si CapsLock y ScrollLock estuvieran activados, nuestra salida sería 80 para ApuntaKb y KeyBrd, los bits 6 y 4 se encuentran en estado alto representando $(01010000)_2$. De esta manera se pueden encontrar varias combinaciones y valores.

Si se deseara conocer el valor del bit de estado de alguna tecla especial, por ejemplo CapsLock, se debe realizar un desplazamiento hacia la derecha dentro del registro hasta colocar el bit de interés como bit 0. Una vez realizado se aplica el operador And con el operando $01.

```
Uses CRT;

Var
      KeyBrd      : Byte Absolute $0040:$0017;
      ValBit      : Byte;
```

```
Begin
  ValBit := KeyBoard Shr $06;                {01000000 a 00000001}
  ValBit := (KeyBoard Shr $06) And $01;      {01000000 a XXXXXXX1}
  WriteLn('ValBit := ', ValBit:1);
  WriteLn('KeyBrd := ', KeyBrd:3);
End.
```

La bandera CapsLock se encuentra activa, por lo cual el bit 6 se encuentra en estado alto, y al realizar el corrimiento de 6 bits a la derecha nos da un valor de 1, aplicando el operador And se limita la salida a un solo bit ignorando los restantes.

Si por alguna razón fuera necesario manejar valores booleanos en lugar de números, se tendrán que realizar algunas modificaciones al programa anterior:

```
Uses CRT;

Var
      KeyBrd      : Byte Absolute $0040:$0017;
      CapsLock    : Boolean;
Begin
  CapsLock := (KeyBrd Shr $06) And $01;          { Valor Numérico }
  CapsLock := ((KeBrd Shr $06) And $01) > 0);    { Valor Booleano }
  If CapsLock Then
    WriteLn('CapsLock Activo')
  Else
    WriteLn('CapsLock DesActivo');
    WriteLn('KeyBrd := ', KeyBrd:3);
End.
```

- ***Cambio de estado de los bits Insert, CapsLock, NumLock y ScrollLock.***

La modificación de alguno de los bits de estado implica la conservación de los estados de los bits restantes, por lo que se utilizará el operador excluyente *OR* (*XOR*). La tabla de verdad de este operador se proporciona a continuación:

$$F := XY' + X'Y = X' \oplus Y'$$

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

El operador XOR requiere de dos operandos, en este caso el primer operando es el byte que contiene el estado del teclado y el segundo se determina con la siguiente formula:

$$Operando2 := 2^{(NumBit)}$$

	NumBit	$2^{(NumBit)}$	Hexadecimal	Binario
ScrollLock	4	16	10h	00010000
NumLock	5	32	20h	00100000
CapsLock	6	64	40h	01000000
Insert	7	128	80h	10000000

De acuerdo a lo anterior, la implementación de este algoritmo se proporciona a continuación:

```
Uses CRT;

Var
      KeyBrd      : Byte Absolute $0040:$0017;
      CapsLock    : Boolean;

Begin
 KeyBrd := KeyBrd Xor $40;          { 01000000 a 00000000 }
 CapsLock := ((KeyBrd Shr $06) And $01) > 0;
 If CapsLock Then
   WriteLn('CapsLock Activo')
 Else
   WriteLn('CapsLock DesActivo');
   WriteLn('KeyBrd := ', KeyBrd:3);
End.
```

En este ejemplo, CapsLock se encuentra activo por lo que al momento de aplicar el operador XOR, se desactiva devolviendo un valor cero (false).

Estado	Hexadecimal	Insert	CapsLock	NumLock	ScrollLock
00000000	00h	Off	Off	Off	Off
00010000	10h	Off	Off	Off	On
00100000	20h	Off	Off	On	Off
00110000	30h	Off	Off	On	On
01000000	40h	Off	On	Off	Off
01010000	50h	Off	On	Off	On
01100000	60h	Off	On	On	Off

Continua....

Estado	Hexadecimal	Insert	CapsLock	NumLock	ScrollLock
01110000	70h	Off	On	On	On
10000000	80h	On	Off	Off	Off
10010000	90h	On	Off	Off	On
10100000	A0h	On	Off	On	Off
10110000	B0h	On	Off	On	On
11000000	C0h	On	On	Off	Off
11010000	D0h	On	On	Off	On
11100000	E0h	On	On	On	Off
11110000	F0h	On	On	On	On

```
Insert      := ((KeyBrd Shr $07) And $01)>0
CapsLock    := ((KeyBrd Shr $06) And $01)>0
NumLock     := ((KeyBrd Shr $05) And $01)>0
ScrollLock  := ((KeyBrd Shr $04) And $01)>0
```

Estado	Hexadecimal	Alt	Ctrl	Left Shift	Right Shift
00000000	00h	Off	Off	Off	Off
00000001	01h	Off	Off	Off	On
00000010	02h	Off	Off	On	Off
00000011	03h	Off	Off	On	On
00000100	04h	Off	On	Off	Off
00000101	05h	Off	On	Off	On
00000110	06h	Off	On	On	Off
00000111	07h	Off	On	On	On
00001000	08h	On	Off	Off	Off
00001001	09h	On	Off	Off	On
00001010	0Ah	On	Off	On	Off
00001011	0Bh	On	Off	On	On
00001100	0Ch	On	On	Off	Off
00001101	0Dh	On	On	Off	On
00001110	0Eh	On	On	On	Off
00001111	0Fh	On	On	On	On

```
Alt          := ((KeyBrd Shr $03) And $01)>0
Ctrl         := ((KeyBrd Shr $02) And $01)>0
Left Shift   := ((KeyBrd Shr $01) And $01)>0
Right Shift  := (KeyBrd And $01)>0
```

*** Características de video:**

El solo conocer el modo inicial de video no nos describe claramente las características que posee nuestro adaptador de video, por lo que es necesario consultar otras direcciones. La primera que analizaremos es la referente al modo de video activo, 0040h:0049h.

-D 0040:0049 L 1

0040:0040 03 I.....P.....

En este caso, el modo inicial de video es el tres. Este número debe ser buscado en la tabla siguiente, dónde se indica el valor para cada uno de los modos de video posibles. La prueba que se está realizando como ejemplo, se encuentra en modo texto de un monitor VGA por lo que tenemos:

Modo de video según BIOS: 03h
Modo de video según tabla: Texto, 80x25, 16 Colores.

- Modos de presentación de video:

Modo	Tipo	Adaptador	Definición	Caja	Caracter	Colores
00h	Texto	CGA[3]	320x200	8x8	40x25	16
		EGA[2,3]	320x350	8x14	40x25	16
		MCGA	320x400	8x16	40x25	16
		VGA[1]	360x400	9x16	40x25	16
01	Texto	CGA	320x200	8x8	40x25	16
		EGA[2]	320x350	8x14	40x25	16
		MCGA	320x400	8x16	40x25	16
		VGA[1]	360x400	9x16	40x25	16
02	Texto	CGA[3]	640x200	8x8	80x25	16
		EGA[2,3]	640x350	8x14	80x25	16
		MCGA	640x400	8x16	80x25	16
		VGA[1]	720x400	9x16	80x25	16
03	Texto	CGA	640x200	8x8	80x25	16
		EGA[2]	640x350	8x14	80x25	16
		MCGA	640x400	8x16	80x25	16
		VGA[1]	720x400	9x16	80x25	16
04	Gráfico	CGA/EGA/MCGA/VGA	320x200	8x8	40x25	4
05	Gráfico	CGA/EGA[3]	320x200	8x8	40x25	4
		MCGA/VGA	320x200	8x8	40x25	4

Continua...

Modo	Tipo	Adaptador	Definición	Caja	Caracter	Colores
06	Gráfico	CGA/EGA/MCGA/VGA	640x200	8x8	80x25	2
07	Texto	MDA/EGA	720/350	9x14	80x25	Mono
		VGA	720x400	9x18	80x25	Mono
08	Gráfico	PC*jr*	160x200	8x8	20x25	16
09	Grafico	PC*jr*	320x200	8x8	40x25	16
0A	Gráfico	PC*jr*	640x200	8x8	80x25	4
0B	Reservado	Reservado				
0C	Reservado	Reservado				
0D	Gráfico	EGA/VGA	320x200	8x8	40x25	16
0E	Gráfico	EGA/VGA	640x200	8x8	80x25	16
0F	Gráfico	EGA/VGA	640x350	8x14	80x25	Mono
10	Gráfico	EGA/VGA	640x350	8x14	80x25	16
11	Gráfico	MCGA/VGA	640x480	8x16	80x30	2
12	Gráfico	VGA	640x480	8x16	80x30	16
13	Gráfico	MCGA/VGA	320x200	8x8	40x25	256

[1] Modo VGA ampliado; de otra forma, el VGA puede emular las características ya sea de CGA o de EGA para este modo.
[2] Modo EGA al conectarse a una pantalla de color ampliada; de otra forma, emula las características de CGA para este modo.
[3] Denota tres tonos de gris.

La siguiente dirección a consultar es la 0040h:004Ah, la cual nos indica el número de columnas desplegables en nuestro monitor.

```
-D 0040:004A L 1

0040:0040                    50                    I.....P.....
```

Número de columnas según BIOS: 50h = 80 Columnas

Para conocer el número de líneas desplegables tenemos que consultar la dirección 0040h:0084h:

-D 0040:0084 L 1

0040:0080 18 ..>....`....P..s

Número de líneas según BIOS: 18h = 24 Líneas
Número de líneas: 25 Líneas

El valor contenido en la dirección antes citada, debe de agregarsele un uno para obtener el valor real de líneas desplegables, debido a que BIOS considera la numeración de las líneas desde 0.

Otro dato que es importante conocer es el tamaño de la página de video, para ello consultamos la dirección 0040h:004Ch, tomando en cuenta que se deberán leer dos bytes:

-D 0040:004C L 2

0040:0040 00 10 .....P.....

Tamaño de la página de video: 4096 Bytes

Si deseamos conocer el ancho del caracter desplegado en la pantalla, debemos consultar la dirección 0040h:0085h.

-D 0040:0085 L 1

0040:0080 10 ..>....`....P..s

Ancho del caracter: 16 pixeles

Además, podemos conocer la dirección del puerto en el cual se encuentra instalado el adaptador, inspeccionando el contenido de la dirección 0040h:0063h (2 bytes), tenemos:

-D 0040:0063 L 2

0040:0060 D4 03 .....)0.....Yy..

Puerto I/O Video: 03D4h

* Características de Adaptadores EGA/VGA.

Para los adaptadores de video EGA/VGA, BIOS proporciona información adicional en las direcciones 0040h:0087h, 0040h:0088h y 0040h:0089.

En la dirección 0040h:0087h se almacena la información referente a los bits de control de video, los cuales quedan definidos como:

Bit 0	0 Trasladar el cursor a los modos 0-3 usando Monitor ECD en 350 líneas. 1 Se Inhibe traslación del cursor
Bit 1	0 Monitor a color ó ECD conectado al adaptador de video. 1 Monitor monocromático conectado al adaptador de video.
Bit 2	1 Espera por monitor
Bit 3	0 Adaptador EGA/VGA Activo.
Bit 4	X No utilizado
Bit 6, 5	Tipo de Memoria instalada en el adaptador $(00)_2$ = 64K $(01)_2$ = 128K $(10)_2$ = 192K $(11)_2$ = 256K
Bit 7	1 Clear RAM Disable 0 Clear RAM Enable

Consultando la dirección:

-D 0040:0087 L 1

0040:0080 60 ..>....`....P..s

Bits de control de video: $(01100000)_2$

Trasladar cursor	***Enable***	**Adaptador EGA/VGA**	***Activo***
Monitor conectado	***Color ó ECD***	**Tipo de memoria**	***256K***
Espera por display	***Enable***	**Limpiar memoria**	***Enable***

Nota: Los valores de *Limpiar memoria* y *Espera por display* deben tomar en cuenta lo siguiente:

0 = Enable
1 = Disable

La dirección 0040h:0088h nos permite conocer la disposición de los bits característicos del conector de video (7, 6, 5, 4), y del switch de opciones (3, 2, 1, 0).

-D 0040:0088 L 1

0040:0080 09 ..>....`....P..s

Bits característicos del conector: $(0000)_2$
Switch de opciones: $(1001)_2$

En la dirección 0040h:0089h encontramos la información de los bits de control de video EGA/VGA:

Bit 7 = 1	200 líneas
Bit 6, 5	Reservado
Bit 4 = 1	400 líneas
Bit 3 = 1	Paleta no cargada
Bit 2 = 1	Monitor monocromático
Bit 1 = 1	Escala de grises
Bit 0 = 1	Reservado

-D 0040:0089 L 1

0040:0080 11 ..>....`....P..s

200 líneas desplegables	Disable
400 líneas desplegables	Enable
Paleta cargada	Enable
Monitor monocromático	Disable
Escala de grises	Enable

• Unidades de disco flexible.

Hasta ahora sólo se ha obtenido el número de unidades de disco flexible instaladas; sin embargo, es necesario conocer otras características.

Por el momento, sólo analizaremos dos direcciones 0040h:008Bh y la 0040h:006Fh, que son correspondientes a la velocidad de datos e información del controlador de diskette.

Analizando la dirección 0040h:008Bh conoceremos la velocidad de datos del drive fijada por el controlador:

Bit 7, 6 Ultima velocidad de datos fijada por el controlador:

$(00)_2$ = 500 Kbits/s
$(01)_2$ = 300 Kbits/s
$(10)_2$ = 250 Kbits/s

Bit 5, 4 Ultima velocidad de paso seleccionada para el diskette

Bit 3, 2 Velocidad de transferencia en la operación inicial:

$(00)_2$ = 500 Kbits/s
$(01)_2$ = 300 Kbits/s
$(10)_2$ = 250 Kbits/s

Bit 1, 0 Reservados

Utilizando DEBUG para accesar a la localidad de memoria:

```
-D 0040:008B L 1

0040:0080                              09                ..>....`....P..s
```

Ultima velocidad fijada por el controlador: 500 Kbits/s
Ultima velocidad de paso seleccionada: 500 Kbits/s
Velocidad de transferencia de datos al inicio: 250 Kbits/s

Para la dirección 0040h:008Fh (información de controlador de disco) tenemos:

Bit 7	Reservado
Bit 6 = 1	Un drive determinado para drive 1
Bit 5 = 1	Drive 1 es multivelocidad
Bit 4 = 1	Drive 1 soporta cambio de línea
Bit 3	Reservado
Bit 2 = 1	Un drive determinado para drive 0
Bit 1 = 1	Drive 0 es multivelocidad
Bit 0 = 1	Drive 0 soporta cambio de línea

Utilizando DEBUG para accesar a la localidad de memoria:

```
-D 0040:008F L 1

0040:0080                              73                ..>....`....P..s
```

Drive 1 (B:)	*Enable*	Drive 0 (A:)	*Enable*
Drive 1 (B:) Multivelocidad	*SI*	Drive 0 (A:) Multivelocidad	*SI*
Drive 1 (B:) Cambio de línea	*SI*	Drive 0 (A:) Cambio de línea	*SI*

*** Identificando el equipo.**

La identificación del equipo consiste en consultar la dirección donde se encuentra el modelo, submodelo y revisión, representados por tres bytes, uno por cada identificador.

Para accesar a esta dirección, es necesario utilizar el programa DEBUG como Shell o interprete, introduciendo el programa que se lista a continuación:

```
MOV AH, C0
INT 15
INT 3
```

Debug provee la instrucción A (assambler), que permite introducir código en lenguaje ensamblador a partir del desplazamiento (OffSet) 100h.

```
-A 100
2453:0100 Mov AH, C0
2453:0102 Int 15
2453:0104 Int 3
2453:0105

-g
AX = 0000   BX = E6F5   CX = 0000   DX = 0000   SP = FFEE   BP = 0000
SI = 0000   DI = 0000   DS = 2453   ES = F000   SS = 2453   CS = 2453
IP = 0104   NV          UP          EI          PL          ZR    NA
PE    NC
2453:0104 CC        INT 3

-D F000:E6F5 L A

        F000:E6F0           08 00 FC-01 00 70 00 00 00 00     .c.....p....A
```

Este programa devolverá en los registros ES y BX el segmento y desplazamiento, respectivamente, donde se encuentra la tabla descriptora del sistema; que consta normalmente de 10 bytes. Aunque en esta tabla se puede consultar otras características del sistema, por el momento sólo consultaremos el modelo, submodelo y revisión de nuestro equipo, para esto es necesario disponer de la siguiente tabla:

ID Byte	Submodelo Byte	Sistema	Fecha	Revisión
FF	00	PC	04/24/81	--
FF	00	PC	10/19/81	--
FF	00	PC	10/27/82	--
FE	00	PC XT	11/08/82	--
FD	00	PC*jr*	06/01/83	--
FC	00	PC AT	01/10/84	--
FC	00	PC AT	06/10/85	01
FC	01	PC AT	11/15/85	00
FC	02	PC XT 286	04/21/86	00
FC	04	PS/2 Modelo 50	02/13/87	00
FC	04	PS/2 Modelo 50 Z	04/18/88	03
FC	05	PS/2 Modelo 60	02/13/87	00
FB	00	PC XT	01/10/86	01
FB	00	PC XT	05/09/86	02
FA	00	PS/2 Modelo 30	09/02/86	00
FA	00	PS/2 Modelo 30	12/12/86	01
F9	00	PC convertible	09/13/85	01
F8	00	PS/2 Modelo 80/16	03/30/87	00
F8	01	PS/2 Modelo 80/20	10/07/87	00

Tabla 5.1 Modelo, Submodelo, Revisión y Byte de Identificación de Sistemas

Consultando la dirección F000:E6F5 obtenida con el programa, tenemos lo siguiente:

```
-D F000:E6F5 L A

F000:E6F0                08 00 FC-01 00 70 00 00 00 00        .c.....p....A
```

La salida del comando Dump (D) presenta sólo 10 bytes, los primeros dos nos proporcionan el número de bytes consecutivos de los que consta la tabla descriptora del sistema, en este caso consta de 8 bytes. A partir del byte número tres se toman los tres bytes siguientes para la tabla descriptora, que representan el modelo del sistema, el submodelo y número de revisión respectivamente.

Modelo: FC
SubModelo: 01
Revisión: 00

Con estos datos, determinamos que el equipo es una PC AT con revisión 0. Si el modelo de su equipo no se encuentra en la tabla 5.1, consulte la tabla adicional que aparece en la siguiente página.

* Modelos ROM BIOS

Descripción	CPU	BUS	ROM BIOS	ID Byte	SubModelo	Rev
PC	8088	ISA/8	04/24/81	FF		
PC	8088	ISA/8	10/19/81	FF		
PC	8088	ISA/8	10/27/81	FF		
PC XT	8088	ISA/8	11/08/82	FE		
PC XT	0888	ISA/8	01/10/86	FB	00	01
PC XT	8088	ISA/8	05/09/86	FB	00	02
PC*jr*	8088	ISA/8	06/01/83	FD		
PC Convertible	80C88	ISA/8	09/13/85	F9	00	00
PS/2 25	8086	ISA/8	06/26/87	FA	01	00
PS/2 30	8086	ISA/8	09/02/86	FA	00	00
PS/2 30	8086	ISA/8	12/12/86	FA	00	01
PS/2 30	8086	ISA/8	02/05/87	FA	00	02
PC AT	286	ISA/16	01/10/84	FC		
PC AT	286	ISA/16	06/10/85	FC	00	01
PC AT	286	ISA/16	11/15/85	FC	01	00
PC XT 286	286	ISA/16	04/21/86	FC	02	00
PS/1	286	ISA/16	12/01/89	FC	0B	00
PS/2 25/286	286	ISA/16	06/28/89	FC	09	02
PS/2 30/286	286	ISA/16	08/25/88	FC	09	00
PS/2 30/286	286	ISA/16	06/28/89	FC	09	02
PS/2 35SX	386SX	ISA/16	03/15/91	F8	19	05
PS/2 35SX	386SX	ISA/16	04/04/91	F8	19	06
PS/2 40SX	386SX	ISA/16	03/15/91	F8	19	05
PS/2 40SX	386SX	ISA/16	04/04/91	F8	19	06
PS/2 L40SX	386SX	ISA/16	02/27/91	F8	23	02
PS/2 50	286	MCA/16	02/13/87	FC	04	00
PS/2 50	286	MCA/16	05/09/87	FC	04	01
PS/2 50z	286	MCA/16	01/28/88	FC	04	02
PS/2 50z	286	MCA/16	04/18/88	FC	04	03
PS/2 55SX	386SX	MCA/16	11/02/88	F8	0C	00
PS/2 57SX	386SX	MCA/16	07/03/91	F8	26	02
PS/2 60	286	MCA/16	02/13/87	FC	05	00
PS/2 65SX	386SX	MCA/16	02/08/91	F8	1C	00
PS/2 70 386	386DX	MCA/32	01/29/88	F8	09	00
PS/2 70 386	386DX	MCA/32	04/11/88	F8	09	02
PS/2 70 386	386DX	MCA/32	12/15/89	F8	09	04
PS/2 70 386	386DX	MCA/32	01/29/88	F8	04	00
PS/2 70 386	386DX	MCA/32	04/11/88	F8	04	02
PS/2 70 386	386DX	MCA/32	12/15/89	F8	04	04
PS/2 70 386	386DX	MCA/32	06/08/88	F8	0D	00
PS/2 70 386	386DX	MCA/32	02/20/89	F8	0D	01
PS/2 70 486	486DX	MCA/32	12/01/89	F8	0D	??
PS/2 70 486	486DX	MCA/32	09/29/89	F8	1B	00
PS/2 80 386	386DX	MCA/32	11/21/89	F8	80	01
PS/2 90 486	486DX	MCA/32		F8	13	00
PS/2 95 486	486DX	MCA/32		F8	16	00

* Fabricante y Fecha de BIOS.

Otra característica importante a conocer es el fabricante y la fecha de BIOS instalado en el equipo, para obtener estos datos basta consultar la dirección F0000:0000 para el nombre de fabricante y la FFFF:0005 para la fecha de BIOS. Lo anterior lo realizamos utilizando el programa DEBUG.

-D F000:0000

```
F000:0000  28 28 63 63 29 29 20 43-43 4F 4F 50 50 59 59 52   ((cc)) CCOOPPYYR
F000:0010  52 49 49 47 47 48 48 54-54 20 31 31 39 39 38 38   RIIGGHHTT 119988
F000:0020  34 34 2C 2C 31 31 39 39-38 38 39 39 41 41 77 77   44,,11998899AAww
F000:0030  61 61 72 72 64 64 20 53-53 6F 6F 66 66 74 74 77   aarrdd SSooffttw
F000:0040  77 61 61 72 72 65 65 20-49 49 6E 6E 63 63 2E 2E   waarree IInncc..
F000:0050  41 41 4C 4C 4C 4C 20 52-52 49 49 47 47 48 48 54   AALLLL RRIIGGHHT
F000:0060  54 53 53 20 52 52 45 45-53 53 45 45 52 52 56 56   TSS RREESSEERRVV
F000:0070  45 45 44 44 28 63 29 20-43 4F 50 59 52 49 47 48   EEDD(c) COPYRIGH
```

Accesando a la dirección F000:0000, obtenemos que el fabricante del BIOS es Award Software Inc ((c) COPYRIGHT 1984, 1989 Award Software Inc. ALL RIGHTS RESERVED).

-D FFFF:0005 L 8

```
FFFF:0000                31 32 2F-30 32 2F 39 33          12/02/93
```

Examinando la dirección FFFF:0005, considerando 8 bytes, tenemos que la fecha del BIOS es Diciembre 02, 1993 (12/02/93).

*** Identificando Tarjeta de Video.**

En ocasiones necesitamos conocer el tipo de tarjeta que esta instalada en nuestro equipo, para ello se requeriría abrir el chasis, desmontar la tarjeta de video y buscar el nombre del fabricante y modo gráfico que soporta; aunque en ocasiones esta información no es visible sobre la tarjeta. Otra forma de conocer estos datos, es utilizando el programa DEBUG con el comando Dump (D), en la dirección de video C000:0000. En esta dirección aparece el nombre del fabricante y el modo gráfico que la tarjeta de video instalada soporta.

-D C000:0000

```
C000:0000  55 AA 40 EB 1C 90 F6 42-4F 41 4B 20 56 47 41 20   U.@....BOAK VGA
C000:0010  42 49 4F 53 2C 20 6E 6F-74 20 66 6F 72 20 49 42   BIOS, not for IB
C000:0020  4D 50 51 52 56 57 1E 06-53 55 2E 8E 1E 06 06 E8   MPQRVW..SU......
C000:0030  09 1C B0 0E E8 3A 1B 80-E4 DE E8 81 1B E4 61 0C   .....:........a.
```

```
C000:0040  3C E6 61 24 C3 E6 61 BA-DE 03 EC 24 E0 3C 40 E8   <.a$..a....$.<@.
C000:0050  CF 00 2E 8E 1E 06 06 E8-E6 1B E8 CC 00 E8 DB 1B   ................
C000:0060  E8 FF 01 F6 06 89 00 01-75 07 F6 06 87 00 02 74   ........u......t
C000:0070  0C BE 52 0A B3 0B B0 00-B9 B4 03 EB 0A BE 92 0A   ..R.............
```

En el despliegue anterior, se presenta como fabricante de la tarjeta *BOAK* y como modo gráfico soportado VGA, además de hacer la anotación que la tarjeta no es compatible con monitores IBM.

En las siguientes páginas se presenta el formato para realizar la descripción del sistema que se esté analizando, la primera es un ejemplo de como se debe llenar y la segunda es un formato para utilizar.

DESCRIPCION DE EQUIPO

Tesis: Diagnóstico de PC's e Impresoras

Byte ID	FC	h	Indique el nombre del equipo
ByteSubModelo	01	h	PC AT Compatible
Revisión	00	h	
Dirección consultada	F000:E6F5	h	

Fabricante BIOS	Award		Comentarios:
Fecha de BIOS	12/02/93		En este caso se consultan dos direcciones: F000:0000 y FFFF:0005
Dirección consultada	FFFF:0005	h	

Fabricante tarjeta de video	BOAK		Comentarios
Modo gráfico soportado	VGA		El nombre del fabricante y modo gráfico aparecen en la dirección C000:0000.
Dirección consultada	C000:0000	h	

Memoria base	639	Kb	Dirección consultada	0040:0013	h

Unidades de disco duro	1		Unidades de disco flexible	2	
Coprocesador Matemático	OK		Puertos seriales	2	
Tarjeta de Mouse *externa*	NO		Adaptadores para impresora	2	
Dirección consultada	0040:0010	h	Valor obtenido	8463	h

Puerto	Dirección Base		Puerto	Dirección Base	
COM1	03F8	h	LPT1	0378	h
COM2	02F8	h	LPT2	0278	h
COM3	0000	h	LPT3	0000	h
COM4	0000	h	LPT4	0000	h
Dirección consultada	0040:0000	h			

Observaciones:

Equipo analizado con procesador Intel 80486 DX2 50 Mhz, controlador VT82C486, VESA Local Bus (VLB), rastreador Logitech Scanman 256 (Addr 280h, IRQ11, DMA 3). Tarjeta adicional de adaptador de impresora.

DESCRIPCION DE EQUIPO

Tesis: Diagnóstico de PC's e impresoras

Byte ID		Indique el nombre del equipo
ByteSubModelo		
Revisión		
Dirección consultada		

Fabricante BIOS		Comentarios:
Fecha de BIOS		
Dirección consultada		

Fabricante tarjeta de video		Comentarios
Modo gráfico soportado		
Dirección consultada		

Memoria base		Dirección consultada	

Unidades de disco duro		Unidades de disco flexible	
Coprocesador Matemático		Puertos seriales	
Tarjeta de Mouse *externa*		Adaptadores para impresora	
Dirección consultada		Valor obtenido	

Puerto	Dirección Base	Puerto	Dirección Base
COM1		LPT1	
COM2		LPT2	
COM3		LPT3	
COM4		LPT4	
Dirección consultada			

Observaciones:

V.2 CONECTORES LOOPBACK SERIAL (DB25, DB9) Y PARALELO (DB25/17)

Este tipo de conectores nos ayudan a conocer si los puertos realmente se encuentran en buen estado. No envían información hacia otros dispositivos (periféricos) ni los reciben de ellos.

Por medio de la computadora se realiza un programa capaz de enviar y recibir caracteres (prueba de LoopBack o Retroalimentación), a distintas velocidades de transmisión (110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 Bauds). Con la ayuda de un programa se realiza el censado de envío y recepción, verificando a cada instante algún tipo de falla, si al inicio de la prueba existe alguna (transmisión o recepción), se debe verificar si están bien colocados los conectores,; si lo están, esto quiere decir que el puerto tiene algún tipo de falla o la computadora cuando fue encendida no reconoció adecuadamente los puertos, en este caso, se deberá apagar y encender la computadora.

Existe un conector (ver Apéndice E) que solo se utiliza para verificación de puertos seriales, con la flexibilidad de que se pueden utilizar puentes (alambres), para realizar las conexiones de un probador de LoopBack, pero con la deficiencia que solo puede ser utilizado para puertos de 25 pines, si se desea realizar la prueba para puertos de 9 pines se debe de usar un adaptador, además de no tener la posibilidad de ser utilizado en alguna prueba para puertos paralelos.

A continuación se presentan los diagramas de conexión (Figura 5.2) de los conectores LoopBack:

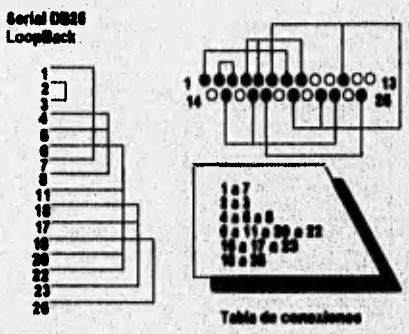


Figura 5.2a

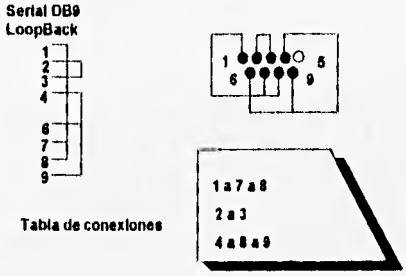


Figura 5.2b

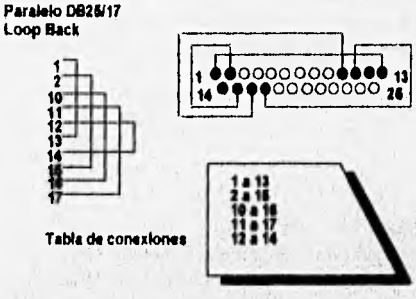


Figura 5.2c

Figura 5.2 Conectores para LoopBack . a) Serial DB25 , b) Serial DB9 y c) Paralelo D25/17

V.3 CONTROL DE OPERACIONES DE IMPRESORA

La utilización del puerto paralelo para manipular la impresora, implica conocer la manera en que DOS accesa a este adaptador o dispositivo de salida. DOS puede accesar simultáneamente cuatro dispositivos de acceso paralelo (0-3), asignando a cada uno de ellos una dirección base, en la cual se almacena un registro de entrada/salida. Para conocer la dirección base de cada uno de los adaptadores es necesario consultar la tabla de datos de BIOS en el segmento 0040:0000. En este segmento se guardan las direcciones base para cada uno de los adaptadores, en localidades de memoria de tipo *word*, inicializandose en cero cuando el adaptador correspondiente no se encuentra disponible.

0040h:0008	Dirección del adaptador LPT1
0040h:000A	Dirección del adaptador LPT2
0040h:000C	Dirección del adaptador LPT3
0040h:000E	Dirección del adaptador LPT4

En la dirección base cada adaptador dispone de tres registros o bytes de control; *registro de salida de datos,* es la dirección del puerto a la cual se debe enviar cada byte de datos en su camino hacia la impresora; el *registro de estado,* permite conocer el estado que guarda la impresora; y el *registro de control,* que inicia y controla la salida de los datos.

Registro de control:

Bit 0 0 = asignación normal, 1 = se produce la salida de un byte
1 0 = asignación normal, 1 = se introduce un salto de línea automático tras cada CR (Carrier Return, retorno de carro)
2 0 = inicia el puerto de impresora, 1 = asignación normal
3 0 = se desactiva la Impresora, 1 = asignación normal
4 0 = se desactiva la interrupción de impresora , 1 = interrupción activada
5-7 no se utilizan

Registro de estado:

Bit 0-2 no se utilizan
3 0 = error de impresora, 1 = no hay error
4 0 = la impresora no está lista, 1 = la impresora está lista
5 0 = la impresora tiene papel, 1 = la impresora no tiene papel
6 0 = la impresora confirma la recepción de caracteres, 1 = normal
7 0 = la impresora está ocupada, 1 = la impresora no está ocupada.

*** Inicialización/reactivación de la impresora**

Los controladores de impresora deben de inicializar cada uno de los puertos paralelos que se utilizarán. La inicialización del puerto paralelo no implica la inicialización de la impresora; sin embargo, este procedimiento es capaz de recobrar la mayoría de los parámetros iniciales de la misma. La reactivación de la impresora después de haber sucedido un error, implica reiniciar la impresora a sus parámetros iniciales y continuar el trabajo un paso antes de la falla, para ello es necesario tener el control de cada una de las acciones del periférico.

La inicialización de la impresora puede ser realizada por medio de la interrupción 17h, servicio 01h, o escribiendo directamente sobre el registro de control. Utilizando la interrupción 17h el programa en lenguaje ensamblador quedaría de la siguiente manera:

```
;Inicializa LPT1:
    MOV AH, 01      ; Función para inicializar la impresora
    MOV DX, 00      ; LPT1
    INT  17h        ; Realiza la inicialización
```

Si se desea inicializar la impresora desde un lenguaje de alto nivel como Turbo Pascal, se utiliza la llamada a interrupción disponible en el mismo, en este caso *Intr*.

```
Program InilPrinter;
Uses CRT, DOS;
Var
      Regs   : Registers;
      Estado : Byte;

Begin
 FillChar(Regs, SizeOf(Regs), 0); {Inicializamos registros en cero}
 With Regs Do
       Begin
         AH := $01;
         DX := $00;
 End;
 Intr($17, Regs);
 Estado := Regs.AH;
End.
```

En este caso, se devuelve el estado de la impresora en el registro AH, que debe ser interpretado bit a bit de acuerdo a:

Bit 0	Tiempo de espera
Bit 1	No se usa
Bit 2	No se usa
Bit 3	Error entrada/salida
Bit 4	Impresora seleccionada
Bit 5	Falta papel
Bit 6	Reconocido
Bit 7	Impresora no ocupada

La interrupción 17h, servicio 01h, básicamente consiste en hacer al bit 2 del registro de control cero durante mil ciclos de un bucle vacio en una AT, ó 1/20 segundos utilizando el contador de reloj del BIOS. En ese momento se requiere que el bit 3, impresora seleccionada, se encuentre activo. Por lo tanto es necesario enviar un 12 al puerto, realizar el retraso, y devolver el valor inicial del registro (08h). A continuación se proporciona el código ensamblador para realizar esta operación:

```
;Inicializa LPT1:
                Mov DX, ES:[8]      ;Pasa la dirección base de DX
                Inc DX              ;Incrementa en uno a DX
                Inc DX              ;Incrementa en uno a DX
                Mov AL, 12          ;Valor de inicialización
                Out DX, AL          ;Empieza la inicialización
Retraso:        Mov AX, 1000        ;Empieza bucle de retraso
                Dec AX              ;Decrementa el contador
                JNZ Retraso         ;realiza el bucle 1000 veces
```

```
Mov AL, 8          ;Valor normal para el registro de control
Out DX, AL         ;Se acaba tiempo, fin de la inicialización.
```

La implementación de este procedimiento en un lenguaje de alto nivel, Turbo Pascal, resulta más sencillo como se observa en el siguiente ejemplo:

```
Program InitPrinter;
Uses CRT, DOS;
Var
      LPT1   : Word Absolute $0040:$0008; { Obtiene dirección base }

Begin
 Port[LPT1+2] := 12;   { Valor de inicialización }
 Delay(1000);
 Port[LPT1+2] := 8;              { Valor inicial del reg. de control}
End.
```

Como primer paso se declara una variable de tipo *word* absoluta que apunta a la tabla de datos del BIOS para obtener la dirección base del puerto LPT1, enseguida se accesa dos localidades arriba de la dirección base al registro de control con la variable Port[ ], asignando a esta localidad el valor de 12, para después dar un retraso de 1/1000 s y escribir el valor inicial del registro 8.

*** Verificación de estado de la impresora**

Una vez inicializada la impresora, esta se encuentra preparada para recibir caracteres. Sin embargo, antes de enviar cualquier caracter a la impresora, es necesario conocer el estado que guarda la misma. Para ello, es necesario monitorear el byte de estado (registro) que se encuentra una localidad arriba de la dirección base del dispositivo.

La operación anterior puede realizarse mediante la interrupción 17h, servicio 02h, ó accesando directamente a la dirección del registro de estado. Utilizando la interrupción 17h, servicio 02h, tenemos lo siguiente:

```
Program GetStPrinter;
Uses CRT, DOS;
Var
      Regs    : Registers;
      Estado  : Byte;

Function GetStatusPRN(NumPRN : Byte) : Byte;

Begin
 FillChar(Regs, SizeOf(Regs), 0);
 With Regs Do
   Begin
     AH := $17;
     DX := NumPRN;  { Número de impresora basado en cero }
```

```
    End;
  GetStatusPRN := Regs.AH;
End;

Begin                    { Principal }
  ClrScr;
  Repeat
    Estado := GetStatusPRN(0);    {Obtener estado de la impresora}
    (***********************************************)
    (* Daspliegue bit a bit del registro de estado *)
    (***********************************************)
    GotoXY(1, 1);
    Write('      Tiempo de espera: ', Estado And $01);
    GotoXY(1, 2);
    Write('Error de Entrada/Salida: ', (Estado Shr $03) And $01);
    GotoXY(1, 3);
    Write(' Impresora seleccionada: ', (Estado Shr $04) And $01);
    GotoXY(1, 4);
    Write('          Falta papel: ', (Estado Shr $05) And $01);
    GotoXY(1, 5);
    Write('     Caracter recibido: ', (Estado Shr $05) And $01);
    GotoXY(1, 6);
    Write('   Impresora no ocupada: ', (Estado Shr $07) And $01);
  Until KeyPrassed;
End.
```

La función GetStatusPRN obtiene el estado actual de la impresora, y lo regresa al programa principal como un valor de tipo byte, este último es asignado a la variable Estado, la cual es desplegada bit a bit dentro de un bucle hasta que se presione alguna tecla. Aunque este método aparenta ser sencillo, puede traer algunas consecuencias para el funcionamiento del programa en el que sea insertado, debido a que se realiza una interrupción un número indefinido de veces, pudiendo ocasionar que en el regreso de la interrupción el programa no pueda continuar su ejecución. Por otro lado, esta función modifica el valor del registro de estado del dispositivo, al hacer cero los tres bits no utilizados en el byte, y realizar una operación XOR con otros dos.

En la siguiente tabla se presentan tres estados posibles representados en el byte de estado. Aunque estos no se pueden generalizar para los diferente modelos y fabricantes de impresoras, pueden tomarse como guías para la interpretación del byte de estado. Quizás no resulte práctico verificar todos los bits de estado de la impresora para escribir sobre ella, por lo que puede limitarse a checar los bits correspondientes a la impresora seleccionada, error de entrada/salida y falta papel.

Valor	Patrón de bits	Significado
144	10010000	Impresora preparada
24	00011000	Impresora no preparada
184	10111000	impresora apagada

La segunda forma de examinar el estado de la impresora, es el consultar directamente el registro de estado del dispositivo. Para ello, es necesario conocer la dirección base del dispositivo, y accesar una localidad arriba de esta:

```
Program GetStPrinter;
Uses CRT, DOS;
Var
      LPT1  : Word Absolute $0040:$0008;  { Obtiene dirección base LPT1 }
      Estado : Byte;

Begin
 Repeat
      Estado := Port[LPT1+1];        { Accesar a registro de Estado }
      GotoXY(1, 1);
      Write('     Tiempo de espera: ', Estado And $01);
      GotoXY(1, 2);
      Write('Error de Entrada/Salida: ', (Estado Shr $03) And $01);
      GotoXY(1, 3);
      Write(' Impresora seleccionada: ', (Estado Shr $04) And $01);
      GotoXY(1, 4);
      Write('          Falta papel: ', (Estado Shr $05) And $01);
      GotoXY(1, 5);
      Write('    Caracter recibido: ', (Estado Shr $05) And $01);
      GotoXY(1, 6);
      Write('  Impresora no ocupada: ', (Estado Shr $07) And $01);
 Until KeyPressed;
End.
```

La interpretación de los bits de estado es diferente con respecto a la que se hace del byte de estado obtenido por medio de la interrupción 17h, en este caso se utiliza el siguiente criterio:

Registro de estado:

Bit	
0-2	no se utilizan
3	0 = error de impresora, 1 = no hay error
4	0 = la impresora no está lista, 1 = la impresora está lista
5	0 = la impresora tiene papel, 1 = la impresora no tiene papel
6	0 = la impresora confirma la recepción de caracteres, 1 = normal
7	0 = la impresora está ocupada, 1 = la impresora no está ocupada.

En el programa descrito sólo se esta solicitando el estado del primer dispositivo LPT1; sin embargo, se puede accesar a los restantes agregando las direcciones base de estos.

```
Program GetStPrinter;
Uses CRT, DOS;
Var
```

```
    LPT1     : Word Absolute $0040:$0008;  { Obtiene dirección base LPT1 }
    LPT2     : Word Absolute $0040:$000A;  { Obtiene dirección base LPT2 }
    LPT3     : Word Absolute $0040:$000C;  { Obtiene dirección base LPT3 }
    LPT4     : Word Absolute $0040:$000E;  { Obtiene dirección base LPT4 }
    Estado   : Byte;
    Estado1  : Byte;

Function GetStatusPRN(AddrPRN : Word) : Byte;
Begin
 GetStatusPRN := Port[AddrPRN + 1];
End;

Begin
 Repeat
   Estado := GetStatusPRN(LPT1);   { Accesar a registro de Estado }
   GotoXY(1, 1);
   Write('      Tiempo de espera: ', Estado And $01);
   GotoXY(1, 2);
   Write('Error de Entrada/Salida: ', (Estado Shr $03) And $01);
   GotoXY(1, 3);
   Write(' Impresora seleccionada: ', (Estado Shr $04) And $01);
   GotoXY(1, 4);
   Write('          Falta papel: ', (Estado Shr $05) And $01);
   GotoXY(1, 5);
   Write('     Caracter recibido: ', (Estado Shr $05) And $01);
   GotoXY(1, 6);
   Write('   Impresora no ocupada: ', (Estado Shr $07) And $01);
   Estado1 := GetStatusPRN(LPT2);
   GotoXY(41, 1);
   Write('      Tiempo de espera: ', Estado1 And $01);
   GotoXY(41, 2);
   Write('Error de Entrada/Salida: ', (Estado1 Shr $03) And $01);
   GotoXY(41, 3);
   Write(' Impresora seleccionada: ', (Estado1 Shr $04) And $01);
       GotoXY(41, 4);
       Write('          Falta papel: ', (Estado1 Shr $05) And $01);
       GotoXY(41, 5);
       Write('     Caracter recibido: ', (Estado1 Shr $05) And $01);
       GotoXY(41, 6);
       Write('   Impresora no ocupada: ', (Estado1 Shr $07) And $01);
 Until KeyPressed;
End.
```

Valor	Patrón de bits	Significado
223	11001111	Impresora preparada
87	01010111	Impresora no preparada
247	11110111	Impresora apagada
119	01110111	Impresora sin papel

Una vez que se conoce el estado de la impresora, y se comprueba que se puede escribir en ella, se procede a la preparación de los datos para ser enviados. La preparación de los datos implica colocar el byte a escribir en el registro de datos del dispositivo (dirección base), fijar el bit cero del byte de control en uno (al menos 1000 ciclos de un bucle vacío), y regresarlo a cero. Esta operación es realizada por la interrupción 17h servicio 00h; sin embargo, los efectos de ejecutar una interrupción en forma consecutiva ya se han comentado, por lo que es preferible hacerlo directamente en la dirección base del dispositivo.

```
Program GetStPrinter;
Uses CRT, DOS;
Var
      LPT1    : Word Absolute $0040:$0008; { Obtiene dirección base LPT1 }
      LPT2    : Word Absolute $0040:$000A; { Obtiene dirección base LPT2 }
      LPT3    : Word Absolute $0040:$000C; { Obtiene dirección base LPT3 }
      LPT4    : Word Absolute $0040:$000E; { Obtiene dirección base LPT4 }
      Estado  : Byte;
      Estado1 : Byte;
      I       : Integer;

Function GetStatusPRN(AddrPRN : Word) : Byte;
Begin
 GetStatusPRN := Port[AddrPRN + 1];
End;
Procedure WriteToPrinter(LPT : Word;
                         Byte2PRN : Byte;
                         Estado : Byte);
Var
      OKPRN : Boolean;

Begin
 Repeat
      If (((Estado Shr $03) And $01) = 0) Then
        Begin
         Port[LPT] := Byte2PRN;
         Port[LPT+2] := Port[LPT+2] XOr $01;
         For I := 1 to 1000 do;
         Port[LPT+2] := Port[LPT+2] XOr $01;
         OKPRN := True;
        End
      Else
        Begin
         Sound(440);  { Emite un sonido de 440 Hz }
         Delay(10);   { durante 10ms             }
         NoSound;     { Apaga bocina             }
         Estado := GetStatusPRN(LPT);
         OKPRN := Flase
        End;
 Until OKPRN;        { Intenta la escritura hasta OKPRN = True }
End;
```

```
Begin
 For I := 50 To 60 Do { Enviar desde Código ASCII 50 hasta 60 }
    Begin
        Estado := GetStatusPRN(LPT1);    { Solicita estado }
        WriteToPrinter(LPT1, I, Estado);    { Escribe sobre dispositivo }
    End;
End.
```

Observese que el valor escrito en el registro de datos del dispositivo es de tipo *ordinal* y no un caracter, además que el bucle vacío se realiza utilizando un ciclo FOR desde uno hasta 1000, agregando ";" después de "DO".

*** Códigos de Control de impresión**

El control de la impresión implica especificar formatos de página, estilos de tipo de letra, etc., para ello se utilizan cadenas ó *códigos de control.* Normalmente estos códigos se envían como secuencias de escape, es decir, inician con el código ASCII 27, aunque existen algunas excepciones. Cuando la secuencia puede variar de tamaño suele terminarse con el código ASCII 0.

Aunque la mayoría de las impresoras utiliza secuencias de escape para su operación, estás son definidas por cada fabricante. Lo anterior hace necesario crear un controlador específico para cada modelo de impresora. En este caso, se tomará como referencia los códigos de control de las impresoras IBM compatibles con el protocolo EPSON.

Código	Función	1	2	3	4	5	6
Movimientos de papel:							
10	Salto de linea	X	X	X	X	X	X
11	Tabulado vertical	X		X	X	X	X
12	Salto de página	X	X	X	X	X	X
13	Retorno de carro	X	X	X	X	X	X
27, 56	Asignar fin de papel	X	X				
27, 57	Cancelar anterior comando	X	X				
27, 66	Asignar tabulados verticales	X		X	X	X	X
27, 66	Borrar tabulador verticales	X		X	X	X	X
27, 68	Activar salto en perforación		X	X	X	X	X
27, 79	Cancelar salto en perforación		X	X	X	X	X
Movimientos de la cabeza de escritura:							
8	Retroceso	X	X	X		X	X
9	Tabulado horizontal	X	X	X	X	X	X
27, 60	Desplazar la cabeza a la izquierda	X	X	X	X	X	X
27, 62	Asignar índice de movimiento horizontal					X	

Código	Función	1	2	3	4	5	6
27, 68	Asignar tabulados horizontales	X	X	X	X	X	X
27, 68	Cancelar tabulados horizontales	X	X	X	X	X	X
27, 77	Justificación automática	X	X				
27, 80	Conectar/Desconectar espaciado proporcional		X				
27, 85	Conectar/Desconectar impresión unidireccional		X	X			
27, 85	Asignar margenes izquierdo y derecho			X			
27, 100	Avanza espacio variable			X			
27, 101	Retroceso espacio variable			X			
Espaciado de línea/caracter							
27, 48	Espaciado de línea	X	X	X		X	X
27, 49	Espaciado de línea	X	X	X		X	X
27, 50	Salto de línea variable	X	X	X	X	X	X
27, 51, n	Salto de línea variable n/216		X				X
27, 53, n	Conecta/Desconecta salto de línea automático			X	X	X	X
27, 65	Salto de línea varible n/72	X	X	X			X
27, 67	Asignar longitud de página	X	X	X	X	X	X
27, 74, n	Salto de línea variable n/216						
Tipo de letra							
11	Conecta 15 caracteres por pulgada	-	-	-	-	-	-
14	Conecta impresión de doble anchura	X	X	X	X	X	X
15	Conecta impresion comprimida	X	X	X	X	X	X
18	Desconecta impresión comprimida/conecta 10 CPI	X	X	X	X	X	X
20	Desconecta impresión de doble anchura	X	X	X	X	X	X
27, 45, n	Conecta/Desconecta subrayado (n = 1, 0)	X	X	X	X	X	X
27, 58	Conecta 12 caracteres por pulgada			X			X
27, 69	Conecta impresión resaltada	X	X	X			X
27, 70	Desconecta impresión resaltada	X	X	X			X
27, 71	Conecta impresión doble pasada	X	X	X			X
27, 72	Desconecta impresión doble pasada	X	X	X			X
27, 83, 0	Conecta impresión superíndice	X	X	X	X	X	X
27, 83, 1	Conecta impresión subíndice	X	X	X	X	X	X
27, 84	Desconecta impresión Superíndice/Subíndice	X	X	X	X	X	X
27, 87, n	Conecta/Desconecta impresión doble anchura	X	X	X	X	X	X
Miscelâneos							
27, 75, n, m	Gráficos 480 puntos		X				X
27, 76, n, m	Gráficos 960 puntos		X				X
27, 89, n, m	Gráficos 960 puntos		X				X
27, 90, n, m	Gráficos 1920 puntos		X				X
7	Timbre	X	X	X	X	X	X

Código	Función	1	2	3	4	5	6
17	Selecciona impresora	X	X	X	X	X	X
19	Deselecciona impresora	X	X	X	X	X	X
24	Limpia buffer	X	X	X	X	X	X
27, 81, n	Deselecciona impresora especifica	X	X	X			X

Impresoras:

1 Matriz *4 Compacta*
2 Gráfica *5 Jet*
3 Color *6 ProPrinter*

[X] = Disponible, [-] = Depende del modelo

Tabla 5.2 Códigos de Control de Impresión

En la tabla anterior, se presentan los códigos de operación para las impresoras IBM/EPSON; sin embargo, no todos son aplicables para los modelos y tipos de impresora; debido a esto utilizaremos los códigos de operación para las impresoras de matriz de puntos, como ejemplo de programación.

*** Programación de impresora de matriz de puntos.**

Como se recordará la manipulación y programación de la impresora implica conocer los códigos de operación y el estado que guarda la misma. Por otro lado, la impresora no proporciona ningún dispositivo ó señal que controle la posición en la que se encuentra la cabeza de impresión, por lo que el programa controlador deberá de prever esta situación, definiendo en forma virtual la posición actual.

Como primer paso, debemos conocer la dirección base de cada uno de los adaptadores de impresora instalados en el equipo, para ello consultamos la tabla descriptora de BIOS a través de variables absolutas. Para almacenar el estado de la impresora necesitaremos una variable de tipo byte, la cual identificaremos como Estado. A continuación insertaremos el procedimiento de obtención de estado de la impresora GetStatusPRN, ya descrito previamente. También se implementará el procedimiento WriteToPrinter, que se utilizará para enviar lo códigos de operación a la impresora, verificando el estado que guarda y una variable apuntador de archivo tipo texto *Text* denominada LPTSet.

```
Program GetStPrinter;
Uses CRT, DOS;
Var
      LPT1    : Word Absolute $0040:$0008; { Obtiene dirección base LPT1 }
      LPT2    : Word Absolute $0040:$000A; { Obtiene dirección base LPT2 }
      LPT3    : Word Absolute $0040:$000C; { Obtiene dirección base LPT3 }
      LPT4    : Word Absolute $0040:$000E; { Obtiene dirección base LPT4 }
      Estado  : Byte;
      Estado1 : Byte;
```

```
        I          : Integer;
        LPTSet  : Text;

Function GetStatusPRN(AddrPRN : Word) : Byte;
Begin
  GetStatusPRN := Port[AddrPRN + 1];
End;

Procedure WriteToPrinter(Var LPTSet  : Text;
                                Str2PRN : String;
                                Estado  : Byte);
Var
        OKPRN : Boolean;
Begin
  Repeat
        If (((Estado Shr $03) And $01) = 0) Then
          Begin
           If Str2PRN[Length(Str2PRN)] = #13 Then
             WriteLn(LPT, Str2PRN)
           Else
             Write(LPT, Str2PRN);
           OKPRN := True;
          End
        Else
          Begin
           Sound(440);  { Emite un sonido de 440 Hz }
           Delay(10);   { durante 10ms             }
           NoSound;     { Apaga bocina             }
           Estado := GetStatusPRN(LPT);
           OKPRN := Flase
          End;
  Until OKPRN;          { intenta la escritura hasta OKPRN = True }
End;

Begin
   WriteToPrinter(LPT1, Str2PRN, Estado);
End.
```

Una vez que se han implementado los procedimientos, funciones y variables necesarias, se procede a abrir el dispositivo de impresión, para esto utilizamos el procedimiento Assign para asignar el apuntador al dispositivo y ReWrite para escribir sobre él.

```
Begin  {Principal}
  Assign(LPTSet, 'LPT1');
  ReWrite(LPTSet);


End.
```

```
        I          : Integer;
        LPTSet  : Text;

Function GetStatusPRN(AddrPRN : Word) : Byte;
Begin
  GetStatusPRN := Port[AddrPRN + 1];
End;

Procedure WriteToPrinter(Var LPTSet  : Text;
                                Str2PRN : String;
                                Estado  : Byte);
Var
        OKPRN : Boolean;
Begin
  Repeat
        If (((Estado Shr $03) And $01) = 0) Then
          Begin
           If Str2PRN[Length(Str2PRN)] = #13 Then
             WriteLn(LPT, Str2PRN)
           Else
             Write(LPT, Str2PRN);
           OKPRN := True;
          End
        Else
          Begin
           Sound(440); { Emite un sonido de 440 Hz }
           Delay(10);  { durante 10ms            }
           NoSound;    { Apaga bocina            }
           Estado := GetStatusPRN(LPT);
           OKPRN := Flase
          End;
  Until OKPRN;        { Intenta la escritura hasta OKPRN = True }
End;

Begin
   WriteToPrinter(LPT1, Str2PRN, Estado);
End.
```

Una vez que se han implementado los procedimientos, funciones y variables necesarias, se procede a abrir el dispositivo de impresión, para esto utilizamos el procedimiento *Assign* para asignar el apuntador al dispositivo y *ReWrite* para escribir sobre él.

```
Begin  {Principal}
  Assign(LPTSet, 'LPT1');
  ReWrite(LPTSet);


End.
```

Una vez abierto el dispositivo de salida se procede a escribir sobre él, por ejemplo, si se desea escribir en la impresora la siguiente frase "EPSON & IBM Printers", se realiza de la siguiente manera:

```
Begin  {Principal}
 Assign(LPTSet, 'LPT1');
 ReWrite(LPTSet);
 Estado := GetStatusPRN(LPT1);   { Solicitamos estado del dispositivo }
 WriteToPrinter(LPTSet, "EPSON & IBM Printers", Estado);


 Close(LPTSet);
End.
```

Ahora utilizaremos algunos comandos de control de impresión, empleados para seleccionar el tipo de letra. Una hoja de 8.5" de ancho permite la colocación de 80 caracteres de tamaño normal; sin embargo, en ocasiones se requiere colocar un número superior ó inferior a éste, resaltar ó distiguir palabras/frases, para conseguirlo se debe cambiar el tipo de letra utilizado.

	Activa	Desactiva
Impresión comprimida	15	18
Impresión de doble anchura	14	20 ó 13
Impresión resaltada	27, 69	27, 70
Impresión con doble pasada	27, 71	27, 72
	Activa	Desactiva
Impresión subrayada	27, 45, 1	27, 45, 0
Impresión Superíndice	27, 83, 0	27, 84
Impresión Subíndice	27, 83, 1	27, 84

Como se puede observar, cada uno de los tipos de impresión tiene un código de activación y otro de desactivación, por lo que a este tipo de de comandos se les conoce como pareados. Un comando pareado se constituye de un código de activación, un objeto y un código de desactivación.

[Cód.Act.] Objeto **[cód.desact.]**

Por ejemplo, si tuvieramos la frase: "Me gusta el trabajo, me fascina. Podría permanecer horas y horas sentado mirando cómo trabajan", y desearamos resaltar "Me gusta el trabajo":

[27, 69]Me gusta el trabajo[27, 70], me fascina...

de esta manera, las primeras cuatro palabras aparecerán resaltadas en la impresión, siendo válido este formato para todos los demás comandos. Ahora utilizando la misma frase e insertando los otros comandos tenemos:

[27, 69]Me gusta el trabajo**[27, 70]**, **[14]**me fascina**[20]**. **[27, 45, 1]**Podría permanecer**[27, 45, 0][27, 83, 0]** horas y horas**[27, 84][27, 83, 1]** sentado mirando**[27, 84]** cómo trabajan.

Utilizando el programa de impresión:

```
Const
        Bold    = #27+#69;
        EndBold = #27+#70;
        DFnt    = #14;
        EndDFnt = #20;
        Super   = #27+#83+#0;
        Sub     = #27+#83+#0;
        EndSS   = #27+#84;
        Sbr     = #27+#45+#1;
        EndSbr  = #27+#45+#0;

Begin  {Principal}
 Assign(LPTSet, 'LPT1');
 ReWrite(LPTSet);
 Estado := GetStatusPRN(LPT1);   { Solicitamos estado del dispositivo }
 WriteToPrinter(LPTSet, Bold, 'Me gusta el trabajo', EndBold, DFnt, 'me fascina'
                EndFnt, '.', Sbr, 'Podria permanecer', ' ', EndSbr, Super,
                        'horas y horas', ' ', EndSS, Sub, 'sentado mirando', EndSS,
                        ' cómo trabajan.');


 Close(LPTSet);
End.
```

La salida en la impresora sería:

Me gusta el trabajo, **me fascina.** Podría permanecer horas y horas sentado mirando cómo trabajan.

V.4 SIMULADOR DE IMPRESORA

La utilización de un puerto paralelo para la transmisión de datos a la impresora, permite que estas sean utilizadas en diferentes equipos de distintas velocidades o tamaño de bus de datos. La interfaz Centronics proporciona 17 señales eléctricas y tierra. Estas señales son: 8 bits de datos, 4 señales de control y 5 de estado:

PIN	DESCRIPCION	DIRECCION
1	[-] STROBE	OUT
2	[+] Data bit 0	OUT
3	[+] Data bit 1	OUT
4	[+] Data bit 2	OUT
5	[+] Data bit 3	OUT
6	[+] Data bit 4	OUT
7	[+] Data bit 5	OUT
8	[+] Data bit 6	OUT
9	[+] Data bit 7	OUT
10	[-] Acknowledge	IN
11	[+] Busy	IN
12	[+] Paper End	IN
13	[+] Select	IN
14	[-] Auto Feed	OUT
15	[-] Error	IN
16	[-] Initialize Printer	OUT
17	[-] Select Input	OUT
18-25	[-] Ground bit 0-7 return	IN

Tabla 5.3 Interfaz Centronics

El acceso a las señales de datos, estado y control se realiza consultando la dirección base del adaptador de impresora. Al conocer la dirección base del adaptador obtenemos las direcciones correspondientes al registro de estado y control:

Dirección de registro de datos = Dirección base
Dirección de registro de estado = Dirección base + 1
Dirección de registro de control = Dirección base + 2

- *Descripción de registros:*

De ESTADO:

Bit 7	Busy	1 = Impresora no ocupada
Bit 6	Acknowledge	0 = Confirmación
Bit 5	Paper End	1 = No tiene papel
Bit 4	Select	1 = On Line
Bit 3	Error	0 = Hay error de E/S
Bit 2	Reservado	0 = Normal
Bit 1	Reservado	0 = Normal
Bit 0	Reservado	0 = Normal

De CONTROL:

Bit 7	Reservado	
Bit 6	Reservado	
Bit 5	Reservado	
Bit 4	Interrupción	1 = Interrupción activa
Bit 3	Activación	0 = Desactivación
Bit 2	Inicialización	0 = Inicializa
Bit 1	CRL automático	1 = CR Line
Bit 0	Data Strobe	1 = Salida de byte

La forma de programación de acceso a estos registros fue descrita anteriormente (Sección V.3). Por otro lado se proporciona el programa SIMULA V1.0 (código fuente y ejecutable), el cual permite manipular la salida de datos a través del puerto de impresora deseado.

Tomando en cuenta esto, se procederá a dar una descripción sencilla de las características, funcionamiento y operación de la interfaz utilizada para observar la salida/entrada de las señales.

En la figura siguiente (Figura 5.3) se muestra el diagrama del circuito utilizado como un simulador de impresora. Este circuito fue diseñado para realizar la verificación de puertos paralelos y el manejo de las señales de estado de la misma forma que las manejaría una impresora.

La interfaz utilizada consiste en un conjunto de buffers encargados de manipular las señales de E/S.

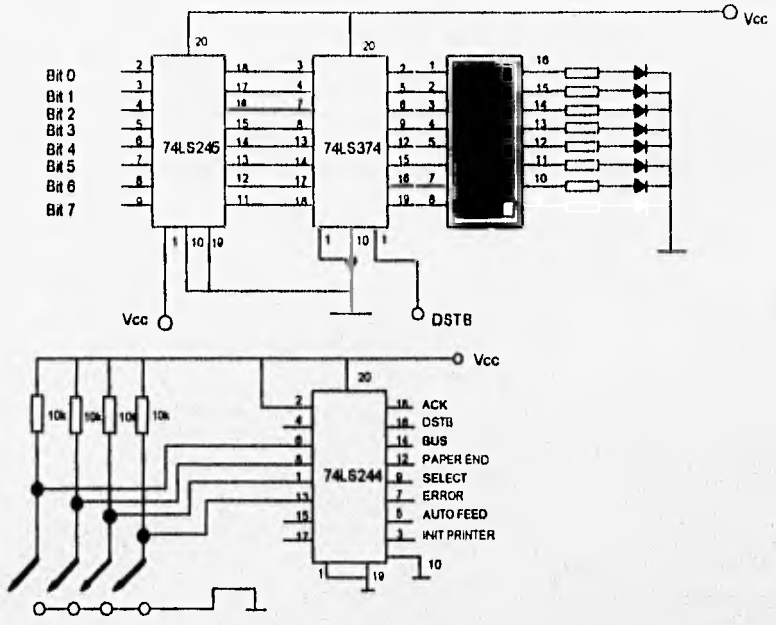


Figura 5.3 Circuito Simulador de Impresora

El circuito de salida (74LS245) utiliza un buffer (Transceiver) para transmitir el valor de las líneas de datos del puerto paralelo de la computadora, a un registro octal tipo D (flip flop) 74LS374 Este registro permitirá la visualización de datos a través de una regleta de LEDs (Bit 0 al 7), esto es realizado al recibir la señal de habilitación de datos (DSTB señal invertida). La visualización del estado de esta señal es mostrada por medio de un LED bicolor y un

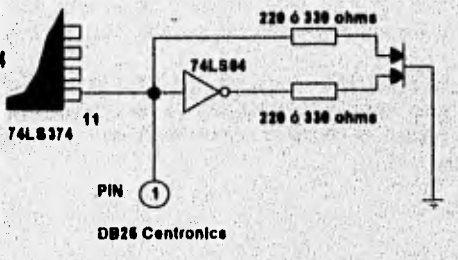


Figura 5.4 Visualización de la señal DSTB a través de un LED bicolor

circuito inversor (74LS04), como se muestra en la Figura 5.4. Esto nos permite conocer los niveles lógicos (1 o 0) que envía la computadora al circuito.

La manipulación de las señales de estado de la impresora se realiza con el buffer octal/linea (74LS244) de control. Las señales Acknowledge, Select y Error deben de estar con valor lógico "1", conectadas al polo positivo (5 V). Las señales Busy y Paper End se colocan en el valor lógico "0", conectadas a tierra. En estas condiciones el circuito es considerado como una impresora seleccionada, sin error, disponible, con reconocimiento de caracteres. La selección de estos niveles lógicos para las señales se obtiene mediante el siguiente circuito (Figura 5.5):

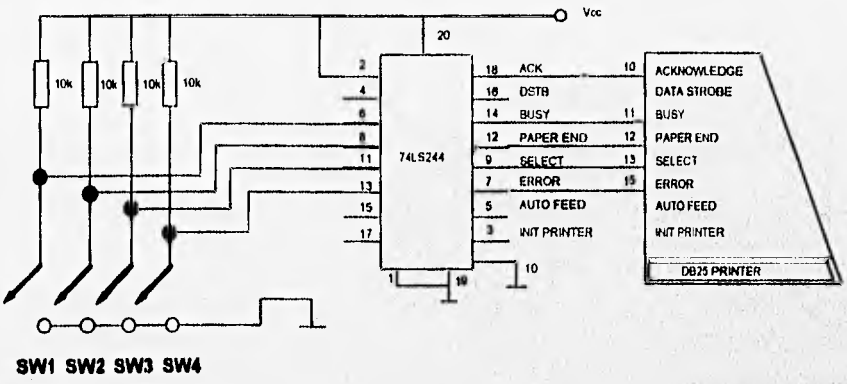


Figura 5.5 Señales de Estado de la Impresora

Al llevar a cabo la simulación del circuito con la ayuda del programa SIMULA V1.0, se envían caracteres al puerto, lo que pondrá brevemente en nivel "1" la salida de control DSTB invertida (esto se verá reflejado en el circuito cuando el led bicolor cambie a rojo), posteriormente se encenderán los leds (bits) que hayan sido solicitados en el programa.

Si por algún motivo se duda del buen funcionamiento del circuito simulador de impresora o del cable conector, se ha diseñado un circuito para realizar pruebas ha dichos dispositivos, este circuito es conocido como *circuito de corrimiento lineal*, mostrado en la siguiente figura (Figura 5.6):

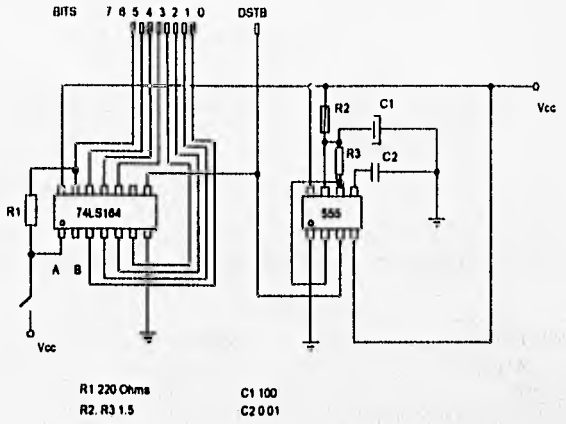


Figura 5.6 Circuito de corrimiento lineal

Como se observa en la figura, este circuito se divide en dos bloques:

1) Mediante un oscilador del tipo 555 en modo astable, se generan pulsos de reloj que modifican la frecuencia de encendido y apagado de las luces en el segundo bloque; si se desea variar dicha frecuencia, puede cambiarse R2 o colocar un potenciómetro.

2) El segundo bloque se encuentra conformado con un registro de corrimiento, utilizando el circuito 74LS164. El registro de corrimiento se encuentra formado por flip-flops conectados en serie, en donde la salida de uno es la entrada del siguiente. En el diseño de este circuito la última salida es conectada a la primera para cerrar la secuencia.

El circuito puede opcionalmente activar o no leds; si estos son habilitados, se puede realizar una comparación con la secuencia que tomen los leds del circuito simulador al ser conectados ambos dispositivos entre sí.

Si los leds del circuito de corrimiento no se encuentran habilitados, el funcionamiento de este circuito se observa en dos leds que nos muestran los niveles alto y bajo de los pulsos enviados por el circuito 555.

V.5 PROGRAMA DE DIAGNOSTICO

El programa diseñado para el diagnóstico de computadoras e impresoras, presenta la información de estos dispositivos mediante las teclas de función siguientes:

[F1]: Ayuda.

Se muestra la mascarilla de las teclas de función con la descripción del trabajo que realiza cada una de ellas.

[F2]: Configuración del Equipo.

Se presenta la información del equipo de cómputo en uso, dando a conocer las siguientes características:

- Coprocesador
- Mouse Card
- Floppy
- Memoria Base, Extendida
- Sistema Operativo
- Disco Boot
- Teclado
- Disco duro
- Puertos
- Identificación

Además de las características anteriores, se realiza la revisión de las variables de ambiente *PATH* y *AUTOEXEC*.

[F3]: Puertos y Comunicaciones.

Con esta opción se da a conocer la dirección en donde se encuentran definidos los puertos del equipo; esta dirección se encuentra en *ROM BIOS*.

Además, se realiza una prueba de *LOOP BACK* enviando información a los puertos seriales verificando su estado.

[F4]: Unidades de Disco.

Se da inicio al análisis de las unidades de disco existentes, dando a conocer las siguientes características:

- Sectores
- Capacidad del disco
- Estado del motor del drive
- Clusters
- Porcentaje de espacio libre
- Tipos de errores

[F5]: Análisis de Video.

Se proporcionan las características del tipo de video que se está utilizando en el sistema, la memoria del adaptador de video y una sección de misceláneos donde se muestra información

tal como: número de columnas y renglones, tamaño de la paleta de video y, si el monitor es capaz, se despliega la paleta de colores.

[F6]: Impresoras.

Esta prueba se realiza solamente en impresoras de matriz de puntos de 9 agujas que puedan trabajar en modo *EPSON*; obteniéndose información del estado del puerto en el que se encuentra seleccionada la impresora; además, se muestran las banderas de estado de la misma.

[F10]: Salir.

Se da finalización a la sesión de trabajo.

A continuación se presenta el listado del programa diseñado para el diagnóstico de **Computadoras e Impresoras**:

```
{                    Program          : Sistema de Diagnóstico
                     Lenguaje de Programación : Turbo Pascal 5.5 ó 6.0
                     Fecha de Inicio  : Mayo 22, 1995.
                     Fecha de Modificación : Febrero 5, 1996.
                     Descripción      : Proporciona información del adaptador de video.
                                        Temporalmente aloja Funciones y procedimientos
                                        para despliegue de marcos y obtención de infor-
                                        mación de unidades de disco, puertos y otras
                                        características.)

Program Sistema_de_Diagnostico;
Uses
  DOS, CRT, Win, SysNum, Comunica, Drives, Equip, Equip2, MseTst, STDPRN;

Const
  DriveChr : Array [0..5] Of Char = ('A', 'B', 'C', 'D', 'E', 'F');
  Activo       : Array [Boolean] Of String[11] = ('DesActivado', 'Activo');
  TypeMon      : Array [Boolean] Of String[13] = ('Color', 'Monocromático');
  VGAAct       : Array [Boolean] Of String[09] = ('Activo', 'No activo');
  Si_No        : Array [Boolean] Of String[02] = ('No', 'Si');
  Off_On       : Array [Boolean] Of String[03] = ('OFF', 'ON');

Var
  AddrHexa     : String;
  L            : Char;
  KeyF         : Char;
  Op           : Char;
  OpCnf        : Char;
  DrvOp        : Char;
  Vdo          : Char;
  KeyHelp      : Char;
  DrvCode      : Byte;
  ReScan       : Boolean;
  DrvCod       : Byte;
  ErrorCodHD   : String;
```

```
  CodErrorHD      : Byte;
  Code            : Byte;
  Ch1             : Byte;
  VideoCtrl       : Byte Absolute   $0040:$0087;
  EGAVGASW        : Byte Absolute   $0040:$0088;
  EGAVGACtrl      : Byte Absolute   $0040:$0089;
  Cols            : Byte Absolute   $0040:$004A;
  Rows            : Byte Absolute   $0040:$0084;
  PageSize        : Word Absolute   $0040:$004C;
  CCursor         : Word Absolute   $0040:$0060;
  PortVideo       : Word Absolute   $0040:$0063;
  HeightChr       : Byte Absolute   $0040:$0085;
  LPT1            : Word Absolute   $0040:$0008;
  LPT2            : Word Absolute   $0040:$000A;
  LPT3            : Word Absolute   $0040:$000C;
  LPT4            : Word Absolute   $0040:$000E;
  StatePRN        : Word;

Procedure ShowScr;
Begin
  SetColors(LightBlue, LightGray);
  CreateWindow(12, 10, 68, 15, S, 'SDPCPRN V1.Beta');
  SetColors(LightBlue, LightGray);
  ClrScr;
  Center('Sistema de Diagnóstico de PC"s e Impresoras', 2);
  Center('Universidad Nacional Autónoma de México', 4);
  SetColors(LightBlue, LightRed);
  Center('Facultad de Ingeniería', 5);
  Repeat
  Until KeyPressed;
  CloseWin(12, 10, 68, 15);
End;

(* Procedimiento de inicio de ambiente del sistema *)
Procedure InitScr;
Begin
  SetWindow(1, 1, 80, 1);
  SetColors(LightGray, Black);
  ClrScr;
  Write('SDPCPRN V1.Beta |      UNAM, Facultad de Ingeniería 1995-96     |');
  Window(1, 2, 80, 24);
  FillWin(#178, LightGray + Black * 16);
End;

Procedure Scrfunc;
Begin
  SetWindow(1, 25, 80, 25);
  SetColors(LightGray, Black);
  ClrScr;
  SetColors(LightGray, Red); Write(' F1 ');
  SetColors(LightGray, Black); Write('Ayuda |');
  SetColors(LightGray, Red); Write(' F2 ');
  SetColors(LightGray, Black); Write('Config ');
  SetColors(LightGray, Red); Write(' F3 ');
  SetColors(LightGray, Black); Write('Comunica ');
  SetColors(LightGray, Red); Write(' F4 ');
  SetColors(LightGray, Black); Write('Disco ');
  SetColors(LightGray, Red); Write(' F5 ');
  SetColors(LightGray, Black); Write('Video ');
  SetColors(LightGray, Red); Write(' F6 ');
  SetColors(LightGray, Black); Write('Impreso |');
```

```
  SetColors(LightGray, Red); Write(' F10 ');
  SetColors(LightGray, Black); Write('Salir');
End;

Procedure KyBrd;
Var
   TMPWord          : Word;
   Code             : Byte;
   Ch1              : Byte;
Begin
  SetColors(LightBlue, LightGray);
  CreateWindow(4, 4, 76, 21, S, 'Estado del Teclado');
  SetColors(LightBlue, LightGray);
  ClrScr;
  WriteKB;
  FrameItem(20, 17, 35, 17, S, LightGreen, 'Código ASCII');
  FrameItem(2, 17, 17, 17, S, LightRed, 'Examinación');
  FrameItem(38, 17, 49, 17, S, LightGreen, 'Caracter');
 Repeat
   GetKeystroke(Code, Ch1);
   LightChar(Code, CH1);
   GotoXY(25, 17); Write(Ch1:3);   { Código ASCII }
   GotoXY( 8, 17); Write(Code:3);  { Código de examinación }
   GotoXY(43, 17);
   If ((Ch1 = 13) Or (Ch1 = 8)) Then
     Write('*')              { Caracter ASCII no desplegable }
   Else
     Write(Chr(Ch1));        { Caracter ASCII desplegable }
 Until (Ch1 = 0) And (Code = 94);
 CloseWin(4, 4, 76, 21);
End;

Procedure TestMouse;
Var
  NumBot            : Byte;
  Xmse, YMse        : Integer;
  HMse              : Word;
Begin
  SetWindow(1, 1, 80, 25);
  NumBot := GetNumBottons;
  SetBoxMouse((3*8)-1, (3*8)-1, (40*8)-1, (21*8)-1);
  SetColors(Black, LightGray);
  CreateWindow(3, 3, 40, 21, d, 'Mouse Pizarra');
  SetColors(Black, LightGray);
  ClrScr;
  SetColors(Black, LightGreen);
  GotoXY(1,1);  Write('(0,0)');
  GotoXY(33,1);  Write('(37,0)');
  GotoXY(1,19);  Write('(0,18)');
  GotoXY(31,19); Write('(37,18)');
  SetColors(Black, LightGray);
  WriteXY(2,  2, 'Coloque el mouse en cada una de las');
  WriteXY(2,  3, 'esquinas de la pizarra y verifique ');
  WriteXY(2,  4, 'las coordenas:                      ');
  WriteXY(2,  6, '    (0,0) ┌──────────────────┐(37, 0)  ');
  WriteXY(2,  7, '          │                  │         ');
  WriteXY(2,  8, '          │                  │         ');
  WriteXY(2,  9, '          │                  │         ');
  WriteXY(2, 10, '          │                  │         ');
  WriteXY(2, 11, '          │                  │         ');
  WriteXY(2, 12, '          │                  │         ');
```

```
  WriteXY(2, 13, '              |                              | ');
  WriteXY(2, 14, '              |                              | ');
  WriteXY(2, 15, '  (0, 18)└──────────────────────────(37, 18) ');
  WriteXY(2, 17, 'Presione CapsLock para desplegar o ');
  WriteXY(2, 18, 'no el apuntador del mouse...       ');
  SetColors(LightBlue, LightGray);
  CreateWindow(45, 3, 76, 7, S, 'Descripción del Mouse');
  ClrScr;
  GotoXY(2, 1); Write('         Tipo: ', GetMouseType);
  GotoXY(2, 4); Write('Versión Driver: ', VermouseDrv:4:2);
  GotoXY(2, 2); Write('          IRQ: ', GetMseIRQ:2);
  GotoXY(2, 3); Write('      Botones: ', NumBot:2);
  GotoXY(2, 5); Write('  Mensajes en: ', GetLggMouse);
  SetColors(LightBlue, LightGray);
  CreateWindow(45, 10, 76, 21, S, 'Información Test');
  SetColors(LightBlue, LightGray);
  ClrScr;
  GotoXY(15,  1); Write('┌──────────────────────┐');
  GotoXY(15,  2); Write('│                      │');
  GotoXY(15,  3); Write('│  Left  │  Right  │');
  GotoXY(15,  4); Write('│        └─────────┘');
  GotoXY(15,  5); Write('│                      │');
  GotoXY(15,  6); Write('│                      │');
  GotoXY(15,  7); Write('│                      │');
  GotoXY(15,  8); Write('│                      │');
  GotoXY(15,  9); Write('└┐                    ┌┘');
  GotoXY(15, 10); Write(' └────────────────────┘');
  ShowMousePtr(True);
  SetCursorSize(7, 0);
  FrameItem(2, 3, 10, 3, S, LightGreen, 'X  Y');
  FrameItem(2, 7, 10, 7, S, LightGreen, ' PTR ');
  GotoXY(3, 9); Write('CapsLock');
  GotoXY(3, 12); Write('Presione SHIFT para salir...');
  Repeat
    GetPosXYStateText(XMse, YMse, HMse);
    GotoXY(3, 3); Write('(',((XMse/8)-2):2:0, ',', ((YMse/8)-2):2:0,')');
    If (HMse And $01) > 0 Then
      Begin
        SetColors(LightBlue, White);
        GotoXY(17, 3); Write('LEFT');
        SetColors(LightBlue, LightGray);
      End
    Else
      Begin
        SetColors(LightBlue, LightGray);
        GotoXY(17, 3); Write('Left');
      End;
    If ((HMse Shr $01) And $01) > 0 Then
      Begin
        SetColors(LightBlue, White);
        GotoXY(24, 3); Write('RIGHT');
        SetColors(LightBlue, LightGray);
      End
    Else
      Begin
        SetColors(LightBlue, LightGray);
        GotoXY(24, 3); Write('Right');
      End;
    If (((CapsLock Shr $06) And $01)>0)  Then
      Begin
        ShowMousePtr(True);
```

```
        GotoXY(3, 7); Write('[redacted]');
        End
      Else
        Begin
          ShowMousePtr(False);
          GotoXY(3, 7); Write('      ');
        End;
    Until ((CapsLock And $01)>0) Or (((CapsLock Shr $01) And $01) >0);
    ShowMousePtr(False);
    CloseWin(3, 3, 40, 21);
    Delay(500);
    CloseWin(45, 3, 76, 6);
    Delay(500);
    CloseWin(45, 10, 76, 21);
    Delay(500);
  End;

  Procedure DiagLoop;
  Begin
    CloseWin(10,5,68,16);
    CloseMSGWin(20, 19, 59, 19);
    MSGWin(20, 23, 59, 23, LightBlue, LightGray, 'Presione ENTER para continuar...');
    SetColors(LightBlue, LightGray);
    CreateWindow(9, 3, 70, 20, s, '[F3]');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(35,2); Write('  Diagrama de conexión para');
    GotoXY(35,3); Write('los conectores de la prueba');
    GotoXY(35,4); Write('de LOOP-BACK');
    FrameItem(3, 2, 31, 17, s, LightGreen, 'Serial DB9 y DB25 LoopBack');
    SetColors(LightBlue, White);
    GotoXY(24,12); Write('DB9');
    GotoXY(20,16); Write('DB25');
    SetColors(LightBlue, LightGray);
    GotoXY(5,2);   Write(' 1 ──┐');
    GotoXY(5,3);   Write(' 2 ─┐ │');
    GotoXY(5,4);   Write(' 3 ─┘ │');
    GotoXY(5,5);   Write(' 4 ──┼─┐');
    GotoXY(5,6);   Write(' 5 ──┼─┼');
    GotoXY(5,7);   Write(' 6 ──┼─┼─┐');
    GotoXY(5, 8);  Write(' 7 ──┘ │ │');
    GotoXY(5,9);   Write(' 8 ────┘ │');
    GotoXY(5,10);  Write('11 ──────┼');
    GotoXY(5,11);  Write('15 ──────┼─┐');
    GotoXY(5,12);  Write('17 ──────┼─┼');
    GotoXY(5,13);  Write('18 ──────┼─┼─┐');
    GotoXY(5,14);  Write('20 ──────┘ │ │');
    GotoXY(5,15);  Write('22 ────────┘ │');
    GotoXY(5,16);  Write('23 ──────────┘');
    GotoXY(5,17);  Write('25 ──────────');
    GotoXY(22,3);  Write(' 1 ─┐');
    GotoXY(22,4);  Write(' 2 ─┼─┐');
    GotoXY(22,5);  Write(' 3 ─┼─┘');
    GotoXY(22,6);  Write(' 4 ─┼──');
    GotoXY(22,7);  Write(' 5   │');
    GotoXY(22,8);  Write(' 6 ──┼─');
    GotoXY(22,9);  Write(' 7 ─┐ │');
    GotoXY(22,10); Write(' 8 ─┘ │');
    GotoXY(22,11); Write(' 9 ────');
    FrameItem(45, 7, 60, 17, s, LightGreen, 'Paralelo');
    SetColors(LightBlue, LightGreen);
```

```
 GotoXY(52,7); Write('DB25/17');
 SetColors(LightBlue, LightGray);
 GotoXY(46,8);   Write(' 1  ┐');
 GotoXY(46,9);   Write(' 2  ┼┐');
 GotoXY(46,10);  Write('10 ┼┼┐');
 GotoXY(46,11);  Write('11 ┼┼┼┐');
 GotoXY(46,12);  Write('12 ┼┼┼┼┐');
 GotoXY(46,13);  Write('13 ┘ │ │ │ │');
 GotoXY(46,14);  Write('14 ──┼─┼─┼─┘');
 GotoXY(46,15);  Write('15 ──┘ │ │');
 GotoXY(46,16);  Write('16 ────┘ │');
 GotoXY(46,17);  Write('17 ──────┘');
 EnterKeyPress ;
 CloseWin(9, 3, 70, 20);
 CloseMSGWin(20, 23, 59, 23);
End;

Procedure Video1;
Begin
  Sonido(800,100);
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F5 ');
  SetColors(LightGray, Black); Write('Video |');
  SetColors(LightGray, Black);
  Write(' Análisis de las caracteristicas de video...          |          ');
  MSGWin(20, 23, 59, 23, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(5, 3, 76, 20, S, 'Caracteristicas de Video');
  ClrScr;
  FrameItem(2, 2, 70, 7, S, LightGreen, 'Caracteristicas');
  FrameItem(50, 3, 69, 3, S, LightRed, 'Memoria Adaptador');
  GotoXY(3, 2);
  Write('    Adaptador instalado: ', Copy(GetTypeMonitor, 1, Pos(' ', GetTypeMonitor)-1));
  GotoXY(3, 3);
  Write(' Monitor según adaptador: ', TypeMon[((VideoCtrl Shr $01) And $01)>0]);
  GotoXY(3, 4);
  Write('       Tipo de monitor: ', Copy(GetTypeMonitor, Pos(' ', GetTypeMonitor)+1, Length(GetTypeMonitor)));
  GotoXY(3, 5);
  Write('   Traslación de cursor: ', VGAAct[(VideoCtrl And $01)>0]);
  GotoXY(3, 6);
  Write('   Modo Actual de Video: ', Copy(GetTypeMonitor, 1, Pos(' ', GetTypeMonitor)-1), '[', VideoMode, ']', ' (',
  GetTypeVideo, ' ', ColsVideo:2, 'x', RowsVideo+1:2, ')');
  GotoXY(3, 7);
  Write('     Lineas rastreadas: ', GetLineScan);
  GotoXY(56, 3);
  Write((((VideoCtrl Shr $05) And $03)*84+64):4, ' Kb');
  GotoXY(56, 5); Write('Clear: ', VGAAct[((VideoCtrl Shr $07) And $01)>0]);
  FrameItem(2, 10, 35, 11, S, LightGreen, 'EGA/VGA Switch');
  GotoXY(3, 10); Write('  Conector: ', Off_On[((EGAVGASW Shr $07) And $01)>0]:3, ' ', Off_On[((EGAVGASW
   Shr $06) And $01)>0]:3, ' ', Off_On[((EGAVGASW Shr $05) And $01)>0]:3, ' ', Off_On[((EGAVGASW Shr $04)
   And $01)>0]);
  GotoXY(3, 11);Write('SW Opciones: ', Off_On[((EGAVGASW Shr $03) And $01)>0]:3, ' ', Off_On[((EGAVGASW
   Shr $02) And $01)>0]:3,' ', Off_On[((EGAVGASW Shr $01) And $01)>0]:3, ' ', Off_On[((EGAVGASW And
   $01))>0]:3);
  FrameItem(2, 14, 35, 17, s, LightGreen, 'EGA/VGA Control');
  GotoXY(3, 14); Write('     Paleta cargada: ', SI_NO[((EGAVGACtrl Shr $03) And $01)=0]);
  GotoXY(3, 15); Write('Monitor Monocromático: ', SI_NO[((EGAVGACtrl Shr $02) And $01)>0]);
  GotoXY(3, 16); Write('   Escala de grises: ', SI_NO[((EGAVGACtrl Shr $01) And $01)>0]);
  GotoXY(3, 17); Write('       Control bits: ', (EGAVGACtrl Shr $07) And $01, (EGAVGACtrl Shr $06) And $01,
                       (EGAVGACtrl Shr $05) And $01, (EGAVGACtrl Shr $04) And $01,
                       (EGAVGACtrl Shr $03) And $01, (EGAVGACtrl Shr $02) And $01,
```

```
                        (EGAVGACtrl Shr $01) And $01, (EGAVGACtrl And $01));
  FrameItem(38, 10, 70, 17, d, LightRed, 'Misceláneos');
  GotoXY(39, 10); Write('          Columnas: ', Cols:4);
  GotoXY(39, 11); Write('          Renglones: ', Rows+1:4);
  GotoXY(39, 12); Write(' Altura de caracter: ', HeightChr:4);
  GotoXY(39, 13); Write(' Tamaño de la página: ', PageSize:4, ' Bytes');
  GotoXY(39, 14); Write('   Puerto Utilizado: ', Dec2Hexa(PortVideo, 4):4, ' h');
  SetCursorSize(7, 0);
  GotoXY(39, 15); Write(' Config. del cursor: ', Dec2Hexa(CCursor, 4):4, ' h');
  GotoXY(39, 16); Write('Emulación Cursor CGA: ', Activo[CursorEnable]);
  If ColorECDAttached Then
    Begin
     SetColors(LightBlue, Yellow);
     GotoXY(39,17); Write('Adapt. Compble. con Color ó ECD');
     SetColors(LightBlue, LightGray);
    End;
 CCursor := $0000;
 EnterKeyPress;
 CloseWin(5, 3, 76, 20);
 CloseMSGWin(20, 22, 59, 22);
End;

Procedure Video2;
Begin
  Sonido(800,100);
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F5 ');
  SetColors(LightGray, Black); Write('Video |');
  SetColors(LightGray, Black);
  Write(' Paleta de colores...                    |          ');
  MSGWin(20, 22, 59, 22, Black, LightBlue, 'Presione ENTER para continuar...');
  SetColors(Black, LightBlue);
  CreateWindow(4, 4, 76, 19, S, 'Colores');
  SetColors(Black, LightBlue);
  ClrScr;
  SetCursorSize(7,0);
  ClrScr;
  A := 0;I := 0;
  For X := 1 to 5 Do
   Begin
    SetColors(Black, X);
    GotoXY(10+X+A, 2); Write(
    GotoXY(10+X+A, 3); Write(
    GotoXY(10+X+A, 4); Write(
    GotoXY(13+X+A,5); Write(X);
    A := A + 10;
   End;
  I := 1; A:=0;
  For X := 6 to 10 Do
   Begin
    SetColors(Black, X);
    GotoXY(10+I+A, 7); Write(
    GotoXY(10+I+A, 8); Write(
    GotoXY(10+I+A, 9); Write(
    GotoXY(13+I+A,10); Write(X);
    A := A + 10;
    I := I + 1;
   End;
  I := 1; A:=0;
  For X := 11 to 15 Do
   Begin
```

```
    SetColors(Black, X);
    GotoXY(10+I+A, 12); Write([redacted]);
    GotoXY(10+I+A, 13); Write([redacted]);
    GotoXY(10+I+A, 14); Write([redacted]);
    GotoXY(13+I+A,15); Write(X);
    A := A + 10;
    I := I + 1;
   End;
 EnterKeyPress;
 CloseWin(4, 4, 76, 19);
 CloseMSGWin(20,22,59,22);
End;

Procedure GetCnf;
Var
  I          : Integer;
  Coms       : Byte;
  LPTs       : Byte;

Function GetDateBIOS(DtBIOS: String): String;
Consl
    MesAA  : Packed Array [1..12] Of String[9] = ('Enero', 'Febrero', 'Marzo',
                                    'Abril', 'Mayo', 'Junio',
                                    'Julio', 'Agosto', 'Septiembre',
                                    'Octubre', 'Noviembre', 'Diciembre');
Var
  TMPVar   : Integer;
  TMPVar1  : Integer;
  TMPVar2  : Integer;
  TMPVar3  : Integer;

Begin
  Val(DtBIOS[1], TMPVar, TMPVar3);
  TMPVar := TMPVar * 10;
  Val(DtBIOS[2], TMPVar1, TMPVar3);
  TMPVar3 := TMPVar + TMPVar1;
  GetDateBIOS := MesAA[TMPVar3] + ' '+ DateBIOS[4]+DateBIOS[5]+', 19'+DateBIOS[7]+DateBIOS[8]+'.';
End;
Begin
 MSGWin(20, 23, 59, 23, LightBlue, LightGray, 'Presione ENTER para continuar...');
 SetColors(LightBlue, LightGray);
 CreateWindow(4, 3, 77, 20, S, 'Caracteristicas del Equipo');
 SetColors(LightBlue, LightGray);
 ClrScr;
 GetEquipment(DiskBoot, Coprocesador, TypeKeyB101, Mouse, NumDrives, AdaptPrinter);
 FrameITEM(2, 3, 18, 3, S, LightRed, 'Coprocesador');  WriteXY(6, 3, Coprocesador);
 FrameITEM(21, 3, 37, 3, S, LightRed, 'Disco BOOT');   WriteXY(26, 3, DiskBoot);
 FrameITEM(41, 3, 55, 3, S, LightRed, 'Mouse Card');   WriteXY(44, 3, Mouse);
 FrameITEM(58, 3, 72, 3, S, LightRed, 'Teclado');      WriteXY(61, 3, TypeKeyB101);
 SetDrive(3);
 FrameITEM(2, 6, 36, 7, S, LightGreen, 'Floppy');
 GotoXY( 3, 6); Write('Drive A : ', GetMediaDrive(0));
 GotoXY( 3, 7); Write('Drive B : ', GetMediaDrive(1));
 FrameItem(40, 6, 56, 7, S, LightGreen, 'Puertos');
 GotoXY(41, 6); Write( 'COM"s: ' , (Equipo Shr $09) And $07:3);
 GotoXY(41, 7); Write( 'LPT"s: ' , (Equipo Shr $0E) And $03:3);
 FrameItem(59, 6, 72, 8, S,  LightGreen, 'GameIO');
 GotoXY(61, 6); Write(Present((Equipo Shr $0C) And $01) >0));
 FrameItem(2, 10, 35, 11, S, LightGreen, 'HDD');
 GotoXY(3, 10); Write('Hard Disk Drive C: ', GetCapacityDrive(3):4:0, ' Mb');
 GotoXY(3, 11); Write('Hard Disk Drive D: ', GetCapacityDrive(4):4:0, ' Mb');
```

```
  Regs.AH:=$88;
  INTR($15, Regs);
  FrameItem(2, 14, 35, 14, S, LightGreen, 'Memoria');
  GotoXY(3, 14); Write('Base: ', MemBase:4, ' Kb    Ext: ', Regs.AX:5, ' Kb');
  FrameItem(2, 17, 35, 17, S, LightGreen, 'Sistema Operativo');
  GotoXY(3, 17); Write('Versión DOS: ', GetVersionDOS/256:2:0, '.', GetVersionDOS And $FF);
  FrameItem(38, 10, 72, 14, D, LightRed, 'Identificación');
  GotoXY(39, 10); Write('    Máquina: ', GetNameMachine);
  GotoXY(39, 11); Write(' Adap. Video: ', Copy(GetTypeMonitor, 1, Pos(' ', GetTypeMonitor) - 1));
  GotoXY(39,12); Write('Tipo Monitor: ', Copy(GetTypeMonitor, Pos(' ', GetTypeMonitor)+1,
   Length(GetTypeMonitor)));
  GotoXY(39, 14); Write('  Fecha BIOS: ', GetDateBIOS(DateBIOS));
  FrameItem(38, 17, 72, 17, S, LightGreen, 'Mouse');
  If GetNumBottons <> 0 Then
    Begin
      GotoXY(39, 17); Write('Consultando controlador...');
      GotoXY(39, 17); Write('Mouse ', GetMouseType, ', IRQ ', GetMseIRQ, ', Versión ',VermouseDrv:4:2);
    End
  Else
    Begin
      GotoXY(39, 17); Write('Mouse no Instalado...');
    End;
  EnterKeyPress;
  CloseWin(4, 3, 77, 20);
  CloseMSGWin(20, 22, 59, 22);
End;

Procedure GetDriveCnf;
Var
  CodeError          : String;
  Ready              : String;
  Seek               : Boolean;
  CtrlDrive          : Boolean;
  ErrorCodig         : String;

Begin
  MSGWin(20, 22, 59, 22, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(4, 4, 76, 19, S, 'Estado de las unidades de disco');
  SetColors(LightBlue, LightGray);
  ClrScr;
  GetOpDrive(Opera, Drive, MotorA, MotorB);
  GetStatusDrive(Ready, Seek, CtrlDrive, ErrorCodig);
  FrameItem(3, 3, 70, 12, S, LightGreen, 'Características');
  GotoXY(4,3);
  SetColors(LightBlue, White);
  Write('          Drive actual : ', Drive);
  SetColors(LightBlue, LightGray);
  GotoXY(4,6);   Write('Ultima operación del Drive ', Drive ,' : ',Opera);
  GotoXY(4,8);   Write('          Estado del Motor A : ', MotorA);
  GotoXY(4,9);   Write('          Estado del Motor B : ', MotorB);
  GotoXY(4,11);  Write('El drive se encuentra ', Ready, ' para alguna operación');
  EnterKeyPress;
  CloseMSGWin(4, 4, 76, 21);
  SetColors(Black, LightGray);
End;

Procedure DrivesCMOS(Drive : Byte);
Begin
  MSGWin(20, 22, 59, 22, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
```

```
  CreateWindow(4, 4, 76, 19, S, 'Estado de las unidades de disco');
  SetColors(LightBlue, LightGray);
  ClrScr;
  FillChar(Regs, SizeOf(Regs),0);
  With Regs Do
    Begin
      AH := $08;
      DL := Drive;
    End;
  Intr($13, Regs);
  Tabla := Ptr(Regs.ES, Regs.DI);
  GotoXY(8,4);  WriteLn('    Primer Byte de especificación: ', Dec2Hexa(Tabla^[0],2):3,'h');
  GotoXY(8,5);  WriteLn('   Segundo Byte de especificación: ', Dec2Hexa(Tabla^[1],2):3,'h');
  GotoXY(8,6);  WriteLn('       Pulsos -> apagar el motor: ', Tabla^[2]:3, ' Pulsos.');
  GotoXY(8,7);  WriteLn('              Bytes por sector: ', GetByteSectorCMOS(Tabla^[3]):4,' Bytes.');
  GotoXY(8,8);  WriteLn('   Velocidad de transf. del drive:  ', DataRateTrns,' Bauds.');
  GotoXY(8,9);  WriteLn('Velocidad de transf. del ctrlador:  ', DataRateCtrl,' Bauds.');
  GotoXY(8,10); WriteLn('Tiempo de colocación de la cabeza: ', Tabla^[9]:3, ' ms');
  GotoXY(8,11); WriteLn('    Tiempo de arranque del motor: ', Tabla^[$0A]:3, ' s/8');
  GotoXY(8,12); WriteLn('         No. de pistas por lado: ', Regs.CH+1:3, ' Pistas.');
  GotoXY(8,13); WriteLn('      No. de sectores por pista: ', Regs.CL:3, ' Sectores.');
  GotoXY(8,14); WriteLn('                No. de lados: ', Regs.DH+1:3, ' Lados.');
  GotoXY(8,15); WriteLn('    No. de unidades consecutivas: ', Regs.DI:3, ' U.');
  GotoXY(8,16); WriteLn('      Tipo de unidad según CMOS: ', GetMediaDrive(Drive));
  Sonido(800,100);
End;

Procedure Drives2(Drive : Byte);
Begin
  Repeat
  MSGWin(20, 23, 59, 23, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(4, 3, 76, 20, O, 'Estado de las unidades de Disco');
  SetColors(LightBlue, LightGray);
  ClrScr;
  GetTime(Hrs, Min, Sec, Sec100);
  TempVar :=  Sec;
  TextColor(LightGreen);
  TextColor(LightGray);
  FillChar(Regs, SizeOf(Regs), 0);
  Regs.AH := $36;
  Regs.DL := Drive;          {<Drive>}
  If (Regs.DL = 1) Then
    Begin
      SetColors(LightBlue, White);
      GotoXY(5,1); Write('Drive: A');
      SetColors(LightBlue, LightGray);
    End;
  If (Regs.DL = 2) Then
    Begin
      SetColors(LightBlue, White);
      GotoXY(5,1); Write('Drive: B');
      SetColors(LightBlue, LightGray);
    End;
  If (Regs.DL = 3) Then
    Begin
      SetColors(LightBlue, White);
      GotoXY(5, 1); Write('Drive: C');
      SetColors(LightBlue, LightGray);
    End;
  MsDos(Regs);
```

```
 SectorsPerCluster := Regs.AX;
 AvailableClusters := Regs.BX;
 BytesPerSector    := Regs.CX;
 TotalClusters     := Regs.DX;
 GotoXY(15,2);  Writeln('  Sectores por cluster: ', SectorsPerCluster:12);
 GotoXY(15,3);  Writeln('  Clusters disponibles: ', AvailableClusters:12);
 GotoXY(15,4);  Writeln('      Bytes por Sector: ', BytesPerSector:12);
 GotoXY(15,5);  Writeln('       Total Clusters: ', TotalClusters:12);
 Disco := ((BytesPerSector*SectorsPerCluster) * TotalClusters);
 GotoXY(15,6);  Writeln('    Capacidad en disco: ', Disco:12, ' Bytes.');
 FreeSpaceDisk := (SectorsPerCluster*BytesPerSector*AvailableClusters);
 GotoXY(15,7);  Writeln(' Espacio libre en Disco: ', FreeSpaceDisk:12:0, ' Bytes.');
If Disco <> 0 Then
Begin
 Porcentaje := ((FreeSpaceDisk * 100)/Disco);
 GotoXY(15,8);   Writeln('      % Libre en disco:     ', Porcentaje:7:2, ' %');
 Writeln; Writeln;
 TextColor(Yellow);
 GotoXY(WhereX+2,WhereY); Writeln('Representacion de espacio en disco:');
 Writeln; TextColor(LightGray);
 Bottom := Round(Porcentaje/5);
 TextColor(LightRed);
 GotoXY(2, Wherey);  Writeln('      0  25  50  75  100');
 GotoXY(10, WhereY);
 Writeln('┌────┴────┴────┴────┐');
 GotoXY(WhereX+7, WhereY);
 TextColor(White);
 TextBackGround(LightGray);
 Writeln('┌────┴────┴────┴────┐');
 GotoXY(WhereX+7, WhereY);
 Writeln('│                   │');
 GotoXY(WhereX+7, WhereY);
 Writeln('└───────────────────┘');
 TextColor(LightGreen);
 GotoXY(WhereX+9, WhereY-2);
 For I := 0 to Bottom do
  Begin
    Write(#219);
    Sound(21000);
    Delay(100);
    NoSound;
  End;
  Writeln;  Writeln;
  TextBackGround(LightBlue);
  TextColor(LightGray);
  If Porcentaje <= 10 then
   Begin
     SetColors(LightBlue, White);
     GotoXY(10,9);   Writeln('El disco se encuentra ocupado en un porcentaje considerable.');
     SetColors(LightBlue, LightGray);
   End;
  GetTime(Hrs, Min, Sec, Sec100);
  TempVar2 := Sec;
  TempVar :=  TempVar2 - TempVar;
  GotoXY(13, 18); Write('Tiempo: ',TempVar:5:0, ' s');
  FrameItem(38, 16, 72, 16, S, LightRed, 'Errores');
  GetStatusDrive(Ready, Seek, CtrlDrive, ErrorCodig);
  ErrorCodHD := GotErrorHD(CodErrorHD);
  If (Drive = 1) Or (Drive = 2) Then
   Begin
     GotoXY(38,16);  Write(ErrorCodig);
```

```
      End
     Else
     If (Drive = 3) Then
      Begin
       GotoXY(38,16);  Write(ErrorCodHD);
      End;
     EnterKeyPress;
     End
    Else
     Begin
      CloseWin(4, 3, 76, 20);
      CloseMSGWin(20, 22, 59, 22);
      Sonido(440, 100);
      Delay(100);
      Sonido(440, 100);
      SetColors(LightRed, LightGray);
      CreateWindow(15, 12, 65, 14,s, 'Error');
      SetColors(LightRed, LightGray);
      ClrScr;
      Write('Sector defectuoso ó la unidad no se encuentra lista');
      Write('Introduzca el disco en la unidad seleccionada...', Regs.AH);
      Write('Presione ENTER para continuar...');
      EnterKeyPress;
     End;
    CloseWin(15, 12, 65, 14);
    Until Disco <> 0;
    CloseWin(4, 3, 76, 20);
    CloseMSGWin(20, 22, 59, 22);
  End;

  Procedure DrvA;
   Begin
     SetWindow(1,25,80,25);
     SetColors(LightGray, LightGreen); Write(' F4 ');
     SetColors(LightGray, Black); Write('Disco |');
     SetColors(LightGray, Black);
     Write(' Analizando el Drive A...                    |          ');
    DrvCode := 0;
    FillChar(Regs, SizeOf(Regs), 0);
     With Regs Do
       Begin
         AH := $02;          { Servicio 02h, INT 13h }
         DL := DrvCode;      { Numero de unidad      }
       End;
     Intr($13, Regs);
   If ((((Regs.AH Shr $07) And $01) > 0) )Then
     Begin
       Sonido(440, 100);
       Delay(100);
       Sonido(440, 100);
       SetColors(LightRed, LightGray);
       CreateWindow(15, 12, 65, 14, O, 'Error');
       SetColors(LightRed, LightGray);
       ClrScr;
       Write('Sector defectuoso ó la unidad no se encuentra lista');
       Write('Introduzca el disco en la unidad seleccionada...', Regs.AH);
       Write('Presione ENTER para continuar...');
       EnterKeyPress;
     End;
     CloseWin(15, 12, 65, 14);
     GetDriveCnf;
```

```
    DrivesCMOS(DrvCode);
    EnterKeyPress;
    Sonido(800,100);
    Drives2(1);
    CloseMSGWin(4, 4, 76, 21);
    CloseWin(10, 18, 70, 22);
  End;

Procedure DrvB;
  Begin
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Analizando el Drive B...                    |        ');
    DrvCode := 1;
    FillChar(Regs, SizeOf(Regs), 0);
    With Regs Do
     Begin
      AH := $02;        { Servicio 02h, INT 13h }
      DL := DrvCode;    { Numero de unidad      }
     End;
    Intr($13, Regs);
    If ((((Regs.AH Shr $07) And $01) > 0) )Then
     Begin
      Sonido(440, 100);
      Delay(100);
      Sonido(440, 100);
      SetColors(LightRed, LightGray);
      CreateWindow(15, 12, 65, 14, 0, 'Error');
      SetColors(LightRed, LightGray);
      ClrScr;
      Write('Sector defectuoso ó la unidad no se encuentra lista');
      Write('Introduzca el disco en la unidad seleccionada...', Regs.AH);
      Write('Presione ENTER para continuar...');
      EnterKeyPress;
     End;
    CloseWin(15, 12, 65, 14);
    GetDriveCnf;
    DrivesCMOS(DrvCode);
    EnterKeyPress;
    Sonido(800,100);
    Drives2(2);
    CloseMSGWin(4, 4, 76, 21);
    CloseWin(10, 18, 70, 22);
  End;

Procedure DrvC;
  Begin
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Analizando el Drive C...                    |        ');
    Sonido(800,100);
    Drives2(3);
    CloseMSGWin(4, 4, 76, 21);
    CloseWin(10, 18, 70, 22);
  End;
```

```
Procedure DrvD;
 Begin
   SetWindow(1,25,80,25);
   SetColors(LightGray, LightGreen); Write(' F4 ');
   SetColors(LightGray, Black); Write('Disco |');
   SetColors(LightGray, Black);
   Write(' Analizando el Drive D...                    |       ');
   Sonido(800,100);
   Drives2(4);
   CloseMSGWin(4, 4, 76, 21);
   CloseWin(10, 18, 70, 22);
 End;

Procedure Text1PRN;
Begin
 SetWindow(1,25,80,25);
 SetColors(LightGray, LightGreen); Write(' F6 ');
 SetColors(LightGray, Black); Write('Impreso |');
 SetColors(LightGray, Black);
 Write(' Nota de aviso de la prueba de impresoras...          |<');
 SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
 Write('> Salir');
 MSGWin(21, 20, 60, 20, LightBlue, LightGray, 'Presione ENTER para continuar...');
 SetColors(LightBlue, LightGray);
 CreateWindow(12, 8, 68, 17, S, 'Impresoras');
 SetColors(LightBlue, LightGray);
 ClrScr;
 GotoXY(2,2); Write('  A continuación se realizará la prueba de impresoras, ');
 GotoXY(2,3); Write('por lo que para su ejecución, en la impresora deberá   ');
 GotoXY(2,4); Write('utilizar papel continuo (Stock), debido a que la prueba');
 GotoXY(2,5); Write('es un poco larga, la prueba también funciona con papel ');
 GotoXY(2,6); Write('bond, pero se tendrán problemas con la prueba  de movi-');
 GotoXY(2,7); Write('miento de hoja.');
 GotoXY(2,8); Write('  Esta prueba sólo puede ejecutarse en impresoras de ');
 GotoXY(2,9); Write('matriz de puntos de 9 agujas en modo EPSON.');
End;

Procedure Text2PRN;
Begin
 SetWindow(1,25,80,25);
 SetColors(LightGray, LightGreen); Write(' F8 ');
 SetColors(LightGray, Black); Write('Impreso |');
 SetColors(LightGray, Black);
 Write(' Nota de aviso de la prueba de impresoras...          |<');
 SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
 Write('> Salir');
 MSGWin(21, 19, 60, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
 SetColors(LightBlue, LightGray);
 CreateWindow(12, 8, 68, 16, S, 'Impresoras');
 SetColors(LightBlue, LightGray);
 ClrScr;
 GotoXY(2,2); Write('  A continuación la prueba que se ejecutará proporciona');
 GotoXY(2,3); Write('información de las banderas de estado que envía la     ');
 GotoXY(2,4); Write('impresora (en este caso cualquier tipo de impresora) a ');
 GotoXY(2,5); Write('la computadora; estas banderas pueden ser como las ');
 GotoXY(2,6); Write('siguientes: ');
 GotoXY(2,7); Write('    la impresora se encuentra encendida, apagada, con');
 GotoXY(2,8); Write('    papel, sin papel, etc.');
End;
```

```
Procedure PRNState;
Begin
 SetColors(LightBlue, LightGray);
 CreateWindow(3, 3, 76, 20, S, 'Impresoras');
 SetColors(LightBlue, LightGray);
 ClrScr;
 FrameItem(3, 2, 20, 2, S, LightGreen, 'Byte de estado');
 GotoXY(2, 4); Write('┌──────────────────────────────┐');
 GotoXY(2, 5); Write('│    Estado del puerto         │');
 GotoXY(2, 6); Write('├──────────────────────────┬───┤');
 GotoXY(2, 7); Write('│ Error en Entrada/Salida  │   │');
 GotoXY(2, 8); Write('│                          ├─┬─┤');
 GotoXY(2, 9); Write('│ Impresora seleccionada   │ │ │');
 GotoXY(2, 10); Write('│                          ├─┼─┤');
 GotoXY(2, 11); Write('│ Papel                    │ │ │');
 GotoXY(2, 12); Write('│                          ├─┼─┤');
 GotoXY(2, 13); Write('│ Reconocido               │ │ │');
 GotoXY(2, 14); Write('│                          ├─┼─┤');
 GotoXY(2, 15); Write('│ Impresora disponible     │ │ │');
 GotoXY(2, 16); Write('└──────────────────────────┴─┴─┘');
 SetColors(LightBlue, White);
 GotoXY(37,2); Write('DIAGNOSTICO ESTANDAR DE IMPRESORAS');
 SetColors(LightBlue, LightGray);
End;

Procedure PRN;
Begin
  SetColors(LightBlue, LightGray);
  GotoXY(8, 2); Write(Dec2Hex(StatePRN, 2,8));
  If (((StatePRN Shr $07) And $01) > 0) Then
    Begin
      GotoXY(30, 15); Write(#254);
      GotoXY(35, 15); Write('                    ');
    End
   Else
    Begin
      GotoXY(30, 15); Write(' ');
      GotoXY(35, 15); Write('No se encuentra disponibie la impresora');
    End;
  If (((StatePRN Shr $06) And $01) > 0) Then
    Begin
      GotoXY(30, 13); Write(#254);
    End
   Else
    Begin
      GotoXY(30, 13); Write(' ');
    End;
  If (((StatePRN Shr $05) And $01) > 0) Then
    Begin
      GotoXY(30, 11); Write(#254);
      GotoXY(43, 11); Write('     Falta papel     ');
    End
  Else
    Begin
      GotoXY(30, 11); Write(' ');
      GotoXY(43, 11); Write('                     ');
    End;
  If (((StatePRN Shr $04) And $01) > 0) Then
    Begin
      GotoXY(30, 9); Write(#254);
      GotoXY(35, 9); Write(' La impresora se encuentra encendida  ');
```

```
    End
   Else
    Begin
     GotoXY(30, 9); Write(' ');
     GotoXY(35, 9); Write('  La impresora se encuentra apagada   ');
    End;
  If (((StatePRN Shr $03) And $01) > 0) Then
    Begin
     GotoXY(30,  7); Write(' ');
     GotoXY(35,  7); Write('                                    ');
    End
   Else
    Begin
     GotoXY(30, 7); Write(#254);
     GotoXY(35, 7);  Write('      No se encuentra en ON-LINE     ');
    End;
 End;

Procedure TestPrinter;
Begin
 Repeat
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F6 ');
  SetColors(LightGray, Black); Write('Impreso | ');
  SetColors(LightGray, Black);
  Write(' Menú de puertos de Impresoras...                    |<');
  SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
  Write('> Salir');
  If (((Equipo Shr $0E) And $03) = 1) Then
   Begin
    CloseWin(30, 9, 50, 15);
    SetColors(LightBlue, LightGray);
    CreateWindow(30, 9, 50, 15, O, 'Puertos');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(6,2); Write('[1]   LPT1');
    GotoXY(6,5); Write('[Esc] Salir');
    GotoXY(6,7); Write('Opción: ');
    Repeat
     OpCnf := UpCase(ReadKey);
    Until OpCnf In ['1', #27];
    Case OpCnf Of
     '1' : Begin
           Sonido(600,100);
           Text1PRN;
           Repeat
            OpCnf := UpCase(ReadKey);
           Until OpCnf In [#27, #13];
           If OpCnf = #27 Then
            Begin
             OpCnf:= #44;
             CloseWin(12,8,68,17);
             CloseMSGWin(19, 20, 61, 20);
            End
           Else
           If OpCnf = #13 Then
            Begin
             SetWindow(1,25,80,25);
             SetColors(LightGray, LightGreen); Write(' F6 ');
             SetColors(LightGray, Black); Write('Impreso | ');
             SetColors(LightGray, Black);
```

```
                Write(' Ejecución de la prueba de impresora...          |       ');
                CloseWin(12,8,68,17);
                CloseMSGWin(19, 20, 61,20);
                PRNState;
                Repeat
                  StatePRN := Port[LPT1+1];
                  PRN;
                  Delay(6500);
                Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
                    And (((StatePRN Shr $05) And $01) = 0)));
                GotoXY(30, 17);Write('Espere un momento...');
                Delay(6000);
                Assign(LST, 'LPT1');
                ReWrite(LST);
                GotoXY(30, 17); Write('                ');
                StartPRN;
                CloseWin(3, 3, 76, 20);
                CloseMSGWin(20, 22, 59, 22);
              End;
        End;
    End;
End;
if (((Equipo Shr $0E) And $03) = 2) Then
  Begin
    CloseWin(30, 9, 50, 15);
    SetColors(LightBlue, LightGray);
    CreateWindow(30, 9, 50, 15, 0, 'Puertos');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(6,2); Write('[1]  LPT1');
    GotoXY(6,3); Write('[2]  LPT2');
    GotoXY(6,5); Write('[Esc] Salir');
    GotoXY(6,7); Write('Opción: ');
    Repeat
      OpCnf := UpCase(ReadKey);
    Until OpCnf in ['1', '2', #27];
    Case OpCnf Of
      '1' : Begin
              Sonido(800,1002);
              Text1PRN;
              Repeat
                OpCnf := UpCase(ReadKey);
              Until OpCnf in [#27, #13];
              If OpCnf = #27 Then
               Begin
                 OpCnf:= #44;
                 CloseWin(12,8,68,17);
                 CloseMSGWin(19, 20, 61, 20);
               End
              Else
              If OpCnf = #13 Then
               Begin
                 SetWindow(1,25,80,25);
                 SetColors(LightGray, LightGreen); Write(' F6 ');
                 SetColors(LightGray, Black); Write('Impreso |');
                 SetColors(LightGray, Black);
                 Write(' Ejecución de la prueba de impresora...          |       ');
                 CloseWin(12,8,68,17);
                 CloseMSGWin(19, 20, 61,20);
                 PRNState;
                 Repeat
```

```
          StatePRN := Port[LPT1+1];
          PRN;
          Delay(5000);
        Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
            And (((StatePRN Shr $05) And $01) = 0)));
        GotoXY(30, 17);Write('Espere un momento...');
        Delay(6000);
        Assign(LST, 'LPT1');
        ReWrite(LST);
        GotoXY(30, 17); Write('                    ');
        StartPRN;
        CloseWin(3, 3, 76, 20);
        CloseMSGWin(20, 22, 59, 22);
       End;
      End;
'2' : Begin
      Sonido(800,100);
      Text1PRN;
      Repeat
       OpCnf := UpCase(ReadKey);
      Until OpCnf In [#27, #13];
      If OpCnf = #27 Then
       Begin
        OpCnf:= #44;
        CloseWin(12,8,68,17);
        CloseMSGWin(19, 20, 61, 20);
       End
      Else
      If OpCnf = #13 Then
       Begin
        SetWindow(1,25,80,25);
        SetColors(LightGray, LightGreen); Write(' F6 ');
        SetColors(LightGray, Black); Write('Impreso |');
        SetColors(LightGray, Black);
        Write(' Ejecución de la prueba de impresora...          |       ');
        CloseWin(12,8,68,17);
        CloseMSGWin(19, 20, 61,20);
        PRNState;
        Repeat
         StatePRN := Port[LPT2+1];
         PRN;
         Delay(5000);
        Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
            And (((StatePRN Shr $05) And $01) = 0)));
        GotoXY(30, 17);Write('Espere un momento...');
        Delay(6000);
        Assign(LST, 'LPT2');
        ReWrite(LST);
        GotoXY(30, 17); Write('                    ');
        StartPRN;
        CloseWin(3, 3, 76, 20);
        CloseMSGWin(20, 22, 59, 22);
       End;
      End;
 End;
End;
If (((Equipo Shr $0E) And $03) = 3) Then
 Begin
  CloseWin(30, 9, 50, 15);
  SetColors(LightBlue, LightGray);
  CreateWindow(30, 9, 50, 15, 0, 'Puertos');
```

```
SetColors(LightBlue, LightGray);
ClrScr;
GotoXY(6,2); Write('[1[  LPT1');
GotoXY(6,3); Write('[2]  LPT2');
GotoXY(6,4); Write('[3]  LPT3');
GotoXY(6,5); Write('[Esc] Salir');
GotoXY(6,7); Write('Opción: ');
Repeat
  OpCnf := UpCase(ReadKey);
Until OpCnf In ['1', '2', '3', #27];
Case OpCnf Of
 '1' : Begin
        Sonido(800,100);
        Text1PRN;
        Repeat
          OpCnf := UpCase(ReadKey);
        Until OpCnf In [#27, #13];
        If OpCnf = #27 Then
         Begin
           OpCnf:= #44;
           CloseWin(12,8,68,17);
           CloseMSGWin(19, 20, 61, 20);
         End
        Else
        If OpCnf = #13 Then
        Begin
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso |');
          SetColors(LightGray, Black);
          Write(' Ejecución de la prueba de impresora...          |       ');
          CloseWin(12,8,68,17);
          CloseMSGWin(19, 20, 61,20);
          PRNState;
          Repeat
            StatePRN := Port[LPT1+1];
            PRN;
            Delay(5000);
          Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
              And (((StatePRN Shr $05) And $01) = 0)));
          GotoXY(30, 17);Write('Espere un momento...');
          Delay(6000);
          Assign(LST, 'LPT1');
          ReWrite(LST);
          GotoXY(30, 17); Write('                    ');
          StartPRN;
          CloseWin(3, 3, 76, 20);
          CloseMSGWin(20, 22, 59, 22);
        End;
 End;
 '2' : Begin
       Sonido(800,100);
       Text1PRN;
       Repeat
         OpCnf := UpCase(ReadKey);
       Until OpCnf In [#27, #13];
       If OpCnf = #27 Then
        Begin
          OpCnf:= #44;
          CloseWin(12,8,68,17);
          CloseMSGWin(19, 20, 61, 20);
```

```
        End
       Else
       If OpCnf = #13 Then
        Begin
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso |');
          SetColors(LightGray, Black);
          Write(' Ejecución de la prueba de impresora...          |          ');
          CloseWin(12,8,68,17);
          CloseMSGWin(19, 20, 61,20);
          PRNState;
          Repeat
            StatePRN := Port[LPT2+1];
            PRN;
            Delay(5000);
          Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
            And (((StatePRN Shr $05) And $01) = 0)));
          GotoXY(30, 17);Write('Espere un momento...');
          Delay(6000);
          Assign(LST, 'LPT2');
          ReWrite(LST);
          GotoXY(30, 17); Write('                    ');
          StartPRN;
          CloseWin(3, 3, 76, 20);
          CloseMSGWin(20, 22, 59, 22);
        End;
End;
'3' : Begin
       Sonido(800,100);
       Text1PRN;
       Repeat
         OpCnf := UpCase(ReadKey);
       Until OpCnf In [#27, #13];
       If OpCnf = #27 Then
        Begin
          OpCnf := #44;
          CloseWin(12,8,68,17);
          CloseMSGWin(19, 20, 61, 20);
        End
       Else
       If OpCnf = #13 Then
        Begin
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso |');
          SetColors(LightGray, Black);
          Write(' Ejecución de la prueba de impresora...          |          ');
          CloseWin(12,8,68,17);
          CloseMSGWin(19, 20, 61,20);
          PRNState;
          Repeat
            StatePRN := Port[LPT3+1];
            PRN;
            Delay(5000);
          Until ((((((StatePRN Shr $03) And $01 ) = 1) And (((StatePRN Shr $04) And $01 ) = 1))
              And (((StatePRN Shr $05) And $01) = 0)));
          Delay(6000);
          Assign(LST, 'LPT3');
          ReWrite(LST);
          GotoXY(30, 17); Write('                    ');
```

```
                StartPRN;
                CloseWin(3, 3, 76, 20);
                CloseMSGWin(20, 22, 59, 22);
               End;
        End;
     End;
   End;
  Until (OpCnf = #27);
  CloseWin(30, 9, 50, 15);
End;

Procedure StateFlagsPrinter;
 Begin
  If (((Equipo Shr $0E) And $03) = 1) Then
   Begin
    CloseWin(30, 9, 50, 15);
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F6 ');
    SetColors(LightGray, Black); Write('Impreso |');
    SetColors(LightGray, Black);
    Write(' Menú de puertos de Impresoras...                    |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(30, 9, 50, 15, 0, 'Puertos');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(6,2); Write('[1]  LPT1');
    GotoXY(6,5); Write('[Esc] Salir');
    GotoXY(6,7); Write('Opción: ');
    Repeat
     OpCnf := UpCase(ReadKey);
    Until OpCnf In ['1', #27];
    Case OpCnf Of
     '1' : Begin
           Sonido(800,100);
           Text2PRN;
           Repeat
            OpCnf := UpCase(ReadKey);
           Until OpCnf In [#27, #13];
          If OpCnf = #27 Then
           Begin
            OpCnf:= #44;
            CloseWin(12,6,68,16);
            CloseMSGWin(21, 19, 60, 19);
           End
          Else
          If OpCnf = #13 Then
           Begin
            CloseWin(12,6,68,16);
            CloseMSGWin(21, 19, 60, 19);
            SetWindow(1,25,80,25);
            SetColors(LightGray, LightGreen); Write(' F6 ');
            SetColors(LightGray, Black); Write('Impreso |');
            SetColors(LightGray, Black);
            Write(' Banderas de estado del puerto de impresora...      |          ');
            MSGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
            PRNState;
            SetColors(LightBlue, White);
            GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
            SetColors(LightBlue, LightGray);
```

```
            Repeat
              StatePRN := Port[LPT1+1];
              PRN;
            Until KeyPressed;
            CloseWin(3, 3, 76, 20);
            CloseMSGWin(15, 23, 65, 23);
          End;
        End;
    End;
End;

If (((Equipo Shr $0E) And $03) = 2) Then
 Begin
  CloseWin(30, 9, 50, 15);
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F6 ');
  SetColors(LightGray, Black); Write('Impreso |');
  SetColors(LightGray, Black);
  Write(' Menú de puertos de Impresoras...                |<');
  SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
  Write('> Salir');
  SetColors(LightBlue, LightGray);
  CreateWindow(30, 9, 50, 15, 0, 'Puertos');
  SetColors(LightBlue, LightGray);
  ClrScr;
  GotoXY(6,2); Write('[1]  LPT1');
  GotoXY(6,3); Write('[2]  LPT2');
  GotoXY(6,5); Write('[Esc] Salir');
  GotoXY(6,7); Write('Opción: ');
  Repeat
    OpCnf := UpCase(ReadKey);
  Until OpCnf In ['1', '2', #27];
  Case OpCnf Of
   '1' : Begin
          Sonido(800,1002);
          Text2PRN;
          Repeat
            OpCnf := UpCase(ReadKey);
          Until OpCnf In [#27, #13];
          If OpCnf = #27 Then
           Begin
            OpCnf:= #44;
            CloseWin(12,6,68,16);
            CloseMSGWin(21, 19, 60, 19);
           End
          Else
          If OpCnf = #13 Then
           Begin
            CloseWin(12,6,68,16);
            CloseMSGWin(21, 19, 60, 19);
            SetWindow(1,25,80,25);
            SetColors(LightGray, LightGreen); Write(' F6 ');
            SetColors(LightGray, Black); Write('Impreso |');
            SetColors(LightGray, Black);
            Write(' Banderas de estado del puerto de impresora...        |        ');
            MSGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
            PRNState;
            SetColors(LightBlue, White);
            GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
            SetColors(LightBlue, LightGray);
            Repeat
```

```
            StatePRN := Port[LPT1+1];
            PRN;
          Until KeyPressed;
          CloseWin(3, 3, 76, 20);
          CloseMSGWin(15, 23, 65, 23);
        End;
      End;
  '2' : Begin
        Sonido(800,100);
        Text2PRN;
        Repeat
          OpCnf := UpCase(ReadKey);
        Until OpCnf in [#27, #13];
        If OpCnf = #27 Then
        Begin
          OpCnf:= #44;
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
        End
        Else
        If OpCnf = #13 Then
        Begin
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso |');
          SetColors(LightGray, Black);
          Write(' Banderas de estado del puerto de impresora...      |      ');
          MBGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
          PRNState;
          SetColors(LightBlue, White);
          GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
          SetColors(LightBlue, LightGray);
          Repeat
            StatePRN := Port[LPT2+1];
            PRN;
          Until KeyPressed;
          CloseWin(3, 3, 76, 20);
          CloseMSGWin(15, 23, 65, 23);
        End;
      End;
    End;
End;
If (((Equipo Shr $0E) And $03) = 3) Then
 Begin
  CloseWin(30, 9, 50, 15);
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F6 ');
  SetColors(LightGray, Black); Write('Impreso |');
  SetColors(LightGray, Black);
  Write(' Menú de puertos de Impresoras...          |<');
  SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
  Write('> Salir');
  SetColors(LightBlue, LightGray);
  CreateWindow(30, 9, 50, 15, 0, 'Puertos');
  SetColors(LightBlue, LightGray);
  ClrScr;
  GotoXY(6,2); Write('[1]  LPT1');
  GotoXY(6,3); Write('[2]  LPT2');
  GotoXY(6,4); Write('[3]  LPT3');
```

```
GotoXY(6,5); Write('[Esc] Salir');
GotoXY(6,7); Write('Opción: ');
Repeat
  OpCnf := UpCase(ReadKey);
Until OpCnf in ['1', '2', '3', #27];
Case OpCnf Of
'1' : Begin
        Sonido(800,100);
        Text2PRN;
        Repeat
          OpCnf := UpCase(ReadKey);
        Until OpCnf in [#27, #13];
        If OpCnf = #27 Then
        Begin
          OpCnf:= #44;
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
        End
        Else
        If OpCnf = #13 Then
        Begin
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso | ');
          SetColors(LightGray, Black);
          Write(' Banderas de estado del puerto de impresora...      |          ');
          MSGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
          PRNState;
          SetColors(LightBlue, White);
          GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
          SetColors(LightBlue, LightGray);
          Repeat
            StatePRN := Port[LPT1+1];
            PRN;
          Until KeyPressed;
          CloseWin(3, 3, 76, 20);
          CloseMSGWin(15, 23, 65, 23);
        End;
      End;
'2' : Begin
        Sonido(800,100);
        Text2PRN;
        Repeat
          OpCnf := UpCase(ReadKey);
        Until OpCnf in [#27, #13];
        If OpCnf = #27 Then
        Begin
          OpCnf:= #44;
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
        End
        Else
        If OpCnf = #13 Then
        Begin
          CloseWin(12,8,68,16);
          CloseMSGWin(21, 19, 60, 19);
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F6 ');
          SetColors(LightGray, Black); Write('Impreso | ');
```

```
            SetColors(LightGray, Black);
            Write(' Banderas de estado del puerto de impresora...        |        ');
            MSGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
            PRNState;
            SetColors(LightBlue, White);
            GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
            SetColors(LightBlue, LightGray);
            Repeat
              StatePRN := Port[LPT2+1];
              PRN;
            Until KeyPressed;
            CloseWin(3, 3, 76, 20);
            CloseMSGWin(15, 23, 65, 23);
          End;
        End;
    '3' : Begin
          Sonido(800,100);
          Text2PRN;
          Repeat
            OpCnf := UpCase(ReadKey);
          Until OpCnf In [#27, #13];
          If OpCnf = #27 Then
           Begin
            OpCnf:= #44;
            CloseWin(12,8,68,16);
            CloseMSGWin(21, 19, 60, 19);
           End
          Else
          If OpCnf = #13 Then
           Begin
            CloseWin(12,8,68,16);
            CloseMSGWin(21, 19, 60, 19);
            SetWindow(1,25,80,25);
            SetColors(LightGray, LightGreen); Write(' F6 ');
            SetColors(LightGray, Black); Write('Impreso |');
            SetColors(LightGray, Black);
            Write(' Banderas de estado del puerto de impresora...        |        ');
            MSGWin(15, 23, 65, 23, LightBlue, LightGray, 'Presione cualquier tecla para salir...');
            PRNState;
            SetColors(LightBlue, White);
            GotoXY(37,2); Write('BANDERAS DE ESTADO DE LA IMPRESORA');
            SetColors(LightBlue, LightGray);
            Repeat
              StatePRN := Port[LPT3+1];
              PRN;
            Until KeyPressed;
            CloseWin(3, 3, 76, 20);
            CloseMSGWin(15, 23, 65, 23);
           End;
          End;
    End;
  End;
End;

(* Inicio del Programa Principal *)
Begin
  SetCursorSize(7, 0);
  InitSCR;
  Scrfunc;
  ShowScr;
  Repeat
```

```
(* Tecla de Ayuda [F1] *)
GetKeystroke(Code, Ch1);
If ((Ch1 = 0) And (Code = 134)) Then
  Begin
    ShowScr;
  End;
If ((Ch1 = 0) And (Code = 59)) Then
      Begin
        Sonido(800,100);
        Repeat
        SetWindow(1,25,80,25);
        SetColors(LightGray, LightGreen); Write(' F1 ');
        SetColors(LightGray, Black); Write('Ayuda |');
        SetColors(LightGray, Black);
        Write(' Mascarilla de ayuda de las teclas de función...        |<');
        SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
        Write('> Salir');
        MSGWin(20, 22, 59, 22, LightBlue, LightGray, 'Presione <ESC> para salir...');
        SetColors(LightBlue, LightGray);
        CreateWindow(4, 4, 76, 19, 0, 'Ayuda');
        SetColors(LightBlue, LightGray);
        ClrScr;
        FrameItem(2, 2, 35, 15, s, LightBlue, '-');
        GotoXY(2,4); SetColors(LightBlue, White); Write('[F1]');
        SetColors(LightBlue, LightGray); Write('  Mascarilla de teclas ');
        GotoXY(2,6); SetColors(LightBlue, White); Write('[F2]');
        SetColors(LightBlue, LightGray); Write('  Configuración del Equipo');
        GotoXY(2,8); SetColors(LightBlue, White); Write('[F3]');
        SetColors(LightBlue, LightGray); Write('  Verificación y comunicación');
        GotoXY(8,9); Write('utilizando los puertos');
        GotoXY(2,11); SetColors(LightBlue, White); Write('[F4]');
        SetColors(LightBlue, LightGray); Write('  Verificación de unidades de');
        GotoXY(8,12); Write('Disco');
        FrameItem(38, 2, 71, 15, s, LightBlue, 'Teclas de Función');
        GotoXY(39,4); SetColors(LightBlue, White); Write('[F5]');
        SetColors(LightBlue, LightGray); Write('  Análisis de Video');
        GotoXY(39,6); SetColors(LightBlue, White); Write('[F6]');
        SetColors(LightBlue, LightGray); Write('  Diagnóstico de Impresoras');
        GotoXY(39,8); SetColors(LightBlue, White); Write('[F10]');
        SetColors(LightBlue, LightGray); Write('  Salir del Programa');
        GotoXY(39,11); Write('  Más información al presionar ');
        GotoXY(39,12); Write('las teclas de función...');
        Repeat
          GetKeyStroke(Code, Ch1);
        Until ((Ch1 = 27) Or ((Code in [59..64]) And (Ch1=0)) Or (Code in [68]));
        If ((Ch1 = 0) And (Code = 59)) Then
          Begin
            SetWindow(1,25,80,25);
            SetColors(LightGray, LightGreen); Write(' F1 ');
            SetColors(LightGray, Black); Write('Ayuda |');
            SetColors(LightGray, Black);
            Write(' Tecla [F1], actualmente en uso...                |          ');
            CloseWin(4, 4, 76, 19);
            CloseMSGWin(4, 4, 76, 21);
            MSGWin(20, 13, 59, 13, LightBlue, LightGray, 'Presione ENTER para continuar...');
            SetColors(LightBlue, LightGray);
            CreateWindow(15, 10, 63, 10, s, '[F1]');
            SetColors(LightBlue, LightGray);
            ClrScr;
            GotoXY(2,1); Write('Mascarilla de ayuda de las teclas de función.');
            EnterKeyPress;
```

```
 End;
If ((Ch1 = 0) And (Code = 60)) Then
 Begin
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F1 ');
  SetColors(LightGray, Black); Write('Ayuda |');
  SetColors(LightGray, Black);
  Write(' Tecla [F2], características del equipo...          |          ');
  CloseWin(4, 4, 76, 19);
  CloseMSGWin(4, 4, 76, 21);
  MSGWin(20, 19, 59, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(10, 5, 68, 16, s, '[F2]');
  SetColors(LightBlue, LightGray);
  ClrScr;
  SetColors(LightBlue, White);
  GotoXY(2,1); Write('[1] Características del Equipo');
  SetColors(LightBlue, LightGray);
  GotoXY(2, 3); Write('  En esta sección, se presenta la información del equipo');
  GotoXY(2, 4); Write('de cómputo en uso, dando a conocer las siguientes carac-');
  GotoXY(2, 5); Write('terísticas: ');
  GotoXY(2, 7); Write('   - Coprocesador              - Disco Boot');
  GotoXY(2, 8); Write('   - Mouse Card                - Teclado');
  GotoXY(2, 9); Write('   - Floppy                    - Disco Duro');
  GotoXY(2,10); Write('   - Memoria Base, Extendida   - Puertos');
  GotoXY(2,11); Write('   - Sistema Operativo         - Identificación');
  SetColors(LightBlue, LightGray);
  EnterKeyPress;
  ClrScr;
  GotoXY(2, 2); Write(' Además de las características anteriores, se realiza la');
  GotoXY(2, 3); Write('revisión de las variables de ambiente PATH y AUTOEXEC.');
  GotoXY(2, 5); SetColors(LightBlue, White); Write('AUTOEXEC: ');
  SetColors(LightBlue, LightGray);
  GotoXY(2, 6); Write('  Son presentados los parámetros que se desean ejecutar');
  GotoXY(2, 7); Write('al inicio de una sesión de trabajo, si es que se definió');
  GotoXY(2, 8); Write('previamente el AUTOEXEC.');
  GotoXY(2,10); SetColors(LightBlue, White); Write('PATH: ');
  SetColors(LightBlue, LightGray);
  GotoXY(2,11); Write('  En esta variable se presenta el registro de directorios.');
  EnterKeyPress;
  ClrScr;
  SetColors(LightBlue, White);
  GotoXY(2, 2); Write('[2] Teclado'); SetColors(LightBlue, LightGray);
  GotoXY(2, 4); Write('  Análisis del funcionamiento del teclado, así como el');
  GotoXY(2, 5); Write('código de las teclas.');
  SetColors(LightBlue, White);
  GotoXY(2, 7); Write('[3] Mouse'); SetColors(LightBlue, LightGray);
  GotoXY(2, 9); Write('  Prueba de verificación del estado del mouse.');
  EnterKeyPress;
  CloseWin(10, 5, 68, 16);
  CloseMSGWin(20, 19, 59, 19);
 End;
If ((Ch1 = 0) And (Code = 61)) Then
 Begin
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F1 ');
  SetColors(LightGray, Black); Write('Ayuda |');
  SetColors(LightGray, Black);
  Write(' Tecla [F3], información del sistema de puertos y comunicaciones... ');
  CloseWin(4, 4, 76, 19);
  CloseMSGWin(4, 4, 76, 21);
```

```
    MSGWin(20, 19, 59, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
    SetColors(LightBlue, LightGray);
    CreateWindow(10, 5, 68, 16, s, '[F3]');
    SetColors(LightBlue, LightGray);
    ClrScr;
    SetColors(LightBlue, White);
    GotoXY(2, 2); Write('Puertos y Comunicaciones');
    SetColors(LightBlue, LightGray);
    GotoXY(2, 5); Write('  Con  esta opción se dan a conocer la dirección en donde');
    GotoXY(2, 6); Write('se encuentran definidos los puertos del equipo; esta');
    GotoXY(2, 7); Write('dirección se encuentra en ROM BIOS.');
    GotoXY(2, 8); Write('  Además se realiza una prueba de LOOP BACK, enviando ');
    GotoXY(2, 9); Write('información a los puertos y verificando su estado.');
    EnterKeyPress;
    DiagLoop;
  End;
 If ((Ch1 = 0) And (Code = 62)) Then
  Begin
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F1 ');
    SetColors(LightGray, Black); Write('Ayuda |');
    SetColors(LightGray, Black);
    Write(' Tecla [F4], características de las unidades de disco...|          ');
    CloseWin(4, 4, 76, 19);
    CloseMSGWin(4, 4, 76, 21);
    MSGWin(20, 19, 59, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
    SetColors(LightBlue, LightGray);
    CreateWindow(10, 5, 68, 16, s, '[F4]');
    SetColors(LightBlue, LightGray);
    ClrScr;
    SetColors(LightBlue, White);
    GotoXY(2, 2); Write('Unidades de Disco');
    SetColors(LightBlue, LightGray);
    GotoXY(2, 4); Write('  Al pulsar la tecla [F4], se da inició al análisis de ');
    GotoXY(2, 5); Write('las unidades de disco existentes.');
    GotoXY(2, 6); Write('  Cuando se analiza algún tipo de floppy, aparece una ');
    GotoXY(2, 7); Write('primer ventana, en la que se dan a conocer el drive ');
    GotoXY(2, 8); Write('actual utilizado, la última operación que se realizó, ');
    GotoXY(2, 9); Write('el estado del motor del drive y el estado inmediato ');
    GotoXY(2,10); Write('después de realizar las operaciones anteriores (LISTO o');
    GotoXY(2,11); Write('ESPERANDO una nueva operación).');
    EnterKeyPress;
    ClrScr;
    GotoXY(2, 2); Write('  En la segunda ventana se da información de las ');
    GotoXY(2, 3); Write('características de la forma de trabajo de los drives,');
    GotoXY(2, 4); Write('además de la corroboración del tipo de unidad que se');
    GotoXY(2, 5); Write('encuentra registrada en CMOS.');
    GotoXY(2, 6); Write('  En la última ventana de información, se realiza un');
    GotoXY(2, 7); Write('análisis de lectura de disco, mostrando información');
    GotoXY(2, 8); Write('del disco al que se le realizó la prueba (Sectores, ');
    GotoXY(2, 9); Write('Clusters, Capacidad del disco, % de espacio libre), ');
    GotoXY(2,10); Write('así como el despliegue de tipos de errores (si es que');
    GotoXY(2,11); Write('sucede alguno)');
    EnterKeyPress;
    ClrScr;
    GotoXY(2, 4); Write('  Cuando se realiza la prueba en un disco duro, solo en');
    GotoXY(2, 5); Write('este caso se despliega la ventana en la que se realiza ');
    GotoXY(2, 6); Write('el análisis de lectura de disco, que es igual que en los');
    GotoXY(2, 7); Write('floppys, se obtiene información de Sectores, Clusters,');
    GotoXY(2, 8); Write('Capacidad de disco, % de espacio libre, etc.');
    EnterKeyPress;
```

```
 End;
If ((Ch1 = 0) And (Code = 63)) Then
 Begin
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F1 ');
  SetColors(LightGray, Black); Write('Ayuda |');
  SetColors(LightGray, Black);
  Write(' Tecla[F5], características de video...          |          ');
  CloseWin(4, 4, 76, 19);
  CloseMSGWin(4, 4, 76, 21);
  MSGWin(20, 19, 59, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(10, 5, 68, 16, s, '[F5]');
  SetColors(LightBlue, LightGray);
  ClrScr;
  SetColors(LightBlue, White);
  GotoXY(2, 1); Write('Análisis de Video');
  SetColors(LightBlue, LightGray);
  GotoXY(2, 3); Write('  En esta sección se proporcionan las características ');
  GotoXY(2, 4); Write('del tipo de video que se está utilizando en el sistema,');
  GotoXY(2, 5); Write('obteniendo además la memoria del adaptador de video que');
  GotoXY(2, 6); Write('se está utilizando, también se muestra una sección de ');
  GotoXY(2, 7); Write('misceláneos, en la que se proporciona más información,');
  GotoXY(2, 8); Write('tal como, número de columnas, renglones, tamaño de la ');
  GotoXY(2, 9); Write('página de video, etc.');
  GotoXY(2,10); Write('  Además si el monitor es capaz, se tiene una opción en');
  GotoXY(2,11); Write('la que se despliega la paleta de colores.');
  EnterKeyPress;
 End;
If ((Ch1 = 0) And (Code = 64)) Then
 Begin
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F1 ');
  SetColors(LightGray, Black); Write('Ayuda |');
  SetColors(LightGray, Black);
  Write(' Tecla [F6], información del diagnóstico de impresoras... |          ');
  CloseWin(4, 4, 76, 19);
  CloseMSGWin(4, 4, 76, 21);
  MSGWin(20, 19, 59, 19, LightBlue, LightGray, 'Presione ENTER para continuar...');
  SetColors(LightBlue, LightGray);
  CreateWindow(10, 5, 68, 16, s, '[F6]');
  SetColors(LightBlue, LightGray);
  ClrScr;
  SetColors(LightBlue, White);
  GotoXY(2, 2); Write('Diagnóstico de Impresoras');
  SetColors(LightBlue, LightGray);
  GotoXY(2, 4); Write('  La prueba de diagnóstico realizada en esta sección se');
  GotoXY(2, 5); Write('ejecuta solamente en impresoras de matriz de puntos de ');
  GotoXY(2, 6); Write('9 agujas que puedan trabajar en modo EPSON.');
  GotoXY(2, 8); Write(' Si la prueba es ejecutada en otro tipo de impresoras ');
  GotoXY(2, 9); Write('con diferentes características a la anterior, el progra-');
  GotoXY(2,10); Write('ma realizará operaciones distintas a las que fue');
  GotoXY(2,11); Write('diseñado.');
  EnterKeyPress;
  ClrScr;
  GotoXY(2, 1); Write('  La primer opción de esta sección (Prueba) ejecuta en sí');
  GotoXY(2, 2); Write('la prueba de diagnóstico.');
  GotoXY(2, 3); Write('  Al realizarse esta prueba de diagnóstico, se obtiene ');
  GotoXY(2, 4); Write('información del estado del puerto en el que se encuentra');
  GotoXY(2, 5); Write('seleccionada alguna impresora.');
  GotoXY(2, 6); Write('  Posteriormente se realiza una prueba en la que se ');
```

```
          GotoXY(2, 7); Write('da a conocer el desempeño de la impresora, mediante la');
          GotoXY(2, 8); Write('cual podemos conocer el funcionamiento adecuado o no de');
          GotoXY(2, 9); Write('la misma.');
          GotoXY(2,10); Write('  Existe una segunda opción (Estado) en la que podemos ');
          GotoXY(2,11); Write('conocer las banderas de estado que envía cualquier tipo');
          GotoXY(2,12); Write('de impresora a la computadora.');
          EnterKeyPress;
         End;
        If ((Ch1 = 0) And (Code = 68)) Then
         Begin
          SetWindow(1,25,80,25);
          SetColors(LightGray, LightGreen); Write(' F1 ');
          SetColors(LightGray, Black); Write('Ayuda |');
          SetColors(LightGray, Black);
          Write(' Tecla [F10], finalización del programa...          |       ');
          CloseWin(4, 4, 76, 19);
          CloseMSGWin(4, 4, 76, 21);
          MSGWin(20, 13, 59, 13, LightBlue, LightGray, 'Presione ENTER para continuar...');
          SetColors(LightBlue, LightGray);
          CreateWindow(15, 10, 63, 10, s, '[F10]');
          SetColors(LightBlue, LightGray);
          ClrScr;
          GotoXY(8,1); Write('Se da termino a la sesión de trabajo.');
          EnterKeyPress;
         End;
        Until (Code = 1);
        CloseMSGWin(4, 4, 76, 21);
        CloseWin(10, 18, 70, 22);
        Scrfunc;
    End;

(* Tecla de Configuración del Equipo [F2] *)
  If ((Ch1 = 0) And (Code = 60)) Then
   Begin
    Sonido(800,100);
    Repeat
     SetWindow(1,25,80,25);
     SetColors(LightGray, LightGreen); Write(' F2 ');
     SetColors(LightGray, Black); Write('Config |');
     SetColors(LightGray, Black);
     Write(' Menú de configuración del equipo...          |<');
     SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
     Write('> Salir');
     SetColors(LightBlue, LightGray);
     CreateWindow(30, 10, 56, 16, 0, 'Configuración');
     SetColors(LightBlue, LightGray);
     ClrScr;
     GotoXY(5,2); Write('[1]  Características');
     GotoXY(5,3); Write('[2]  Teclado');
     GotoXY(5,4); Write('[3]  Mouse');
     GotoXY(5,5); Write('[Esc] Salir');
     GotoXY(5,7); Write('Opción: ');
     Repeat
      OpCnf := UpCase(ReadKey);
     Until OpCnf In ['0', '1', '2', '3', #27];
     Case OpCnf Of
       '1' : Begin
              SetWindow(1,25,80,25);
              SetColors(LightGray, LightGreen); Write(' F2 ');
              SetColors(LightGray, Black); Write('Config |');
              SetColors(LightGray, Black);
```

```
      Write(' Opción [1], características de la configuración del equipo...    ');
      Sonido(800, 100);
      GetCnf;
      Autoexec;
      CloseWin(2, 3, 76, 19);
    End;
  '2' : Begin
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F2 ');
      SetColors(LightGray, Black); Write('Config |');
      SetColors(LightGray, Black);
      Write(' Opción [2], se ejecuta la prueba de teclado...      |          ');
      Sonido(800,100);
      KeybText1;
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F2 ');
      SetColors(LightGray, Black); Write('Config |');
      SetColors(LightGray, Black);
      Write(' Simulación del Teclado y prueba de diagnóstico...  |<');
      SetColors(LightGray, Red); Write('CtrlF1'); SetColors(LightGray, Black);
      Write('> Salir');
      KyBrd;
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F2 ');
      SetColors(LightGray, Black); Write('Config |');
      SetColors(LightGray, Black);
      Write(' Prueba en teclas Alt; Shft; Ctrl; Caps, Num, Scroll Lock...|    ');
      Caps;
      CloseWin(2, 3, 76, 19);
      RestoreKeyBrd;
    End;
  '3' : Begin
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F2 ');
      SetColors(LightGray, Black); Write('Config |');
      SetColors(LightGray, Black);
      Write(' Opción [3], diagnóstico del Mouse...          |          ');
      Sonido(800, 100);
      If GetNumBottons <> 0 Then
        Begin
          CloseWin(30, 10, 58, 16);
          SetColors(LightBlue, LightGray);
          CreateWindow(10, 12, 70, 12, S, 'Espere un momento');
          ClrScr;
          GotoXY(12, 1); Write('Inicializando Mouse ', GetMouseType, ' Versión ',
          VermouseDrv:4:2);
          TestMouse;
        End
      Else
        Begin
          SetColors(Red, LightGray);
          Sonido(440, 100);
          Delay(100);
          Sonido(440, 100);
          CreateWindow(10, 8, 70, 16, S, 'Mouse Version 1.1');
          SetColors(Red, LightGray);
          ClrScr;
          GotoXY(2, 2); Write('No se ha detectado el dispositivo apuntador, posiblemente');
          GotoXY(2, 3); Write('no se encuentra conectado ó el controlador del mismo no se');
          GotoXY(2, 4); Write('cargo previamente. ');
          GotoXY(2, 6); Write('Verifique su  presencia  utilizando la  opción  número 1 ');
```

```
                GotoXY(2, 7); Write('del menú actual (Características), ó utilice MOUSE.EXE.');
                GotoXY(27,9); Write('Presione ENTER para continuar...');
                EnterKeyPress;
                CloseMSGWin(20, 19, 59, 19);
                CloseWin(10, 8, 70, 16);
              End;
          End;
      End;
    RestoreKeyBrd;
    Until (OpCnf = #27) Or (OpCnf = '0');
    CloseWin(30, 10, 58, 16);
    Scrfunc;
  End;

(* Tecla de Comunicaciones y Puertos [F3] *)
If ((Ch1 = 0) And (Code = 61)) Then
 Begin
  Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F3 ');
    SetColors(LightGray, Black); Write('Disco ');
    SetColors(LightGray, Black);
    Write(' Menú de diagnóstico de Puertos y Comunicaciones...     ');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(29, 10, 53, 15, 0, 'Comunicaciones');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Características');
    GotoXY(4,3); Write('[b] Loop-Back');
    GotoXY(4,4); Write('[Esc] Salir');
    GotoXY(4,6); Write('Opción: ');
    Repeat
     DrvOp := UpCase(ReadKey);
    Until DrvOp In ['A','B', #27];
    Case DrvOp Of
      'A' : Comunica1;
      'B' : Begin
              TextCom1;
              Repeat
                OpCnf := UpCase(ReadKey);
              Until OpCnf In [#27, #13];
              If OpCnf = #27 Then
                Begin
                  OpCnf:= #44;
                  CloseWin(4, 4, 76, 19);
                  CloseMSGWin(20, 17, 59, 17);
                End
              Else
                If OpCnf = #13 Then
                  Begin
                    CloseWin(4, 4, 76, 19);
                    CloseMSGWin(20, 17, 59, 17);
                    Comunica2;
                  End;
            End;
        End;
  Until (DrvOp = #27);
  CloseWin(29, 10, 53, 15);
  Scrfunc;
```

```
 End;

(* Tecla de Diagnóstico de Unidades de Disco (F4) *)
 If ((Ch1 = 0) And (Code = 62)) Then
 Begin
    Sonido(800,100);
 If (((GetMediaDrive(0) <> 'No instalada') And (GetMediaDrive(1) = 'No instalada')) And
    ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) = 0))) Then
  Begin
    Repeat
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F4 ');
      SetColors(LightGray, Black); Write('Disco |');
      SetColors(LightGray, Black);
      Write(' Menú de diagnóstico de unidades de disco...          |<');
      SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
      Write('> Salir');
      SetColors(LightBlue, LightGray);
      CreateWindow(32, 8, 48, 15, 0, 'Drives');
      SetColors(LightBlue, LightGray);
      ClrScr;
      GotoXY(4,2); Write('[a] Drive A ');
      GotoXY(4,6); Write('[Esc] Salir');
      GotoXY(6,8); Write('Opción: ');
      Repeat
        DrvOp := UpCase(ReadKey);
      Until DrvOp in ['A', #27];
      Case DrvOp Of
        'A' : DrvA;
      End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
 End;
 If (((GetMediaDrive(0) = 'No instalada') And (GetMediaDrive(1) <> 'No instalada')) And
    ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) = 0))) Then
  Begin
    Repeat
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F4 ');
      SetColors(LightGray, Black); Write('Disco |');
      SetColors(LightGray, Black);
      Write(' Menú de diagnóstico de unidades de disco...          |<');
      SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
      Write('> Salir');
      SetColors(LightBlue, LightGray);
      CreateWindow(32, 8, 48, 15, 0, 'Drives');
      SetColors(LightBlue, LightGray);
      ClrScr;
      GotoXY(4,2); Write('[a] Drive B ');
      GotoXY(4,6); Write('[Esc] Salir');
      GotoXY(6,8); Write('Opción: ');
      Repeat
        DrvOp := UpCase(ReadKey);
      Until DrvOp in ['A', #27];
      Case DrvOp Of
        'A' : DrvB;
      End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 If (((GetMediaDrive(0) = 'No instalada') And (GetMediaDrive(1) = 'No instalada')) And
```

```
  ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) <> 0))) Then
 Begin
  Repeat
   SetWindow(1,25,80,25);
   SetColors(LightGray, LightGreen); Write(' F4 ');
   SetColors(LightGray, Black); Write('Disco | ');
   SetColors(LightGray, Black);
   Write(' Menú de diagnóstico de unidades de disco          |');
   SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Blue);
   Write('> Salir');
   SetColors(LightBlue, LightGray);
   CreateWindow(32, 8, 48, 15, 0, 'Drives');
   SetColors(LightBlue, LightGray);
   ClrScr;
   GotoXY(4,2); Write('[a] Drive C ');
   GotoXY(4,6); Write('[Esc] Salir');
   GotoXY(6,8); Write('Opción: ');
   Repeat
    DrvOp := UpCase(ReadKey);
   Until DrvOp in ['A', #27];
   Case DrvOp Of
    'A': DrvC;
   End;
 Until (DrvOp = #27);
 CloseWin(32, 8, 48, 15);
End;
If ((GetUnidadDrive(3) = 'No instalada') And (GetUnidadDrive(4) = 'No instalada')) And
  ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) <> 0)) Then
 Begin
  Repeat
   SetWindow(1,25,80,25);
   SetColors(LightGray, LightGreen); Write(' F4 ');
   SetColors(LightGray, Black); Write('Disco | ');
   SetColors(LightGray, Black);
   Write(' Menú de diagnóstico de unidades de disco          |');
   SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Blue);
   Write('> Salir');
   SetColors(LightBlue, LightGray);
   CreateWindow(32, 8, 48, 15, 0, 'Drives');
   SetColors(LightBlue, LightGray);
   ClrScr;
   GotoXY(4,2); Write('[a] Drive C ');
   GotoXY(4,6); Write('[Esc] Salir');
   GotoXY(6,8); Write('Opción: ');
   Repeat
    DrvOp := UpCase(ReadKey);
   Until DrvOp in ['A', #27];
   Case DrvOp Of
    'A': DrvC;
   End;
 Until (DrvOp = #27);
 CloseWin(32, 8, 48, 15);
End;
If ((GetUnidadDrive(3) = 'No instalada') And (GetUnidadDrive(4) <> 'No instalada')) And
  ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) <> 0)) Then
 Begin
  Repeat
   SetWindow(1,25,80,25);
   SetColors(LightGray, LightGreen); Write(' F4 ');
   SetColors(LightGray, Black); Write('Disco | ');
   SetColors(LightGray, Black);

 92
```

```
  ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) = 0))) Then
 Begin
  Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Menú de diagnóstico de unidades de disco...          |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(32, 8, 48, 15, 0, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive C ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp In ['A', #27];
    Case DrvOp Of
      'A' : DrvC;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
 End;
 If (((GetMediaDrive(0) = 'No instalada') And (GetMediaDrive(1) = 'No instalada')) And
   ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) <> 0))) Then
  Begin
   Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Menú de diagnóstico de unidades de disco...          |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(32, 8, 48, 15, 0, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive D ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp In ['A', #27];
    Case DrvOp Of
      'A' : DrvD;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 If (((GetMediaDrive(0) <> 'No instalada') And (GetMediaDrive(1) <> 'No instalada')) And
   ((GetCapacityDrive(3) = 0) And (GetCapacityDrive(4) = 0))) Then
  Begin
   Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
```

```
    Write(' Menú de diagnóstico de unidades de disco...          |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(32, 8, 48, 15, O, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive A ');
    GotoXY(4,3); Write('[b] Drive B ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp In ['A', 'B', #27];
    Case DrvOp Of
      'A' : DrvA;
      'B' : DrvB;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 If (((GetMediaDrive(0) <> 'No instalada') And (GetMediaDrive(1) = 'No instalada')) And
   ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) = 0))) Then
 Begin
   Repeat
     SetWindow(1,25,80,25);
     SetColors(LightGray, LightGreen); Write(' F4 ');
     SetColors(LightGray, Black); Write('Disco |');
     SetColors(LightGray, Black);
     Write(' Menú de diagnóstico de unidades de disco...          |<');
     SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
     Write('> Salir');
     SetColors(LightBlue, LightGray);
     CreateWindow(32, 8, 48, 15, O, 'Drives');
     SetColors(LightBlue, LightGray);
     ClrScr;
     GotoXY(4,2); Write('[a] Drive A ');
     GotoXY(4,3); Write('[b] Drive C ');
     GotoXY(4,6); Write('[Esc] Salir');
     GotoXY(6,8); Write('Opción: ');
     Repeat
       DrvOp := UpCase(ReadKey);
     Until DrvOp In ['A', 'B', #27];
     Case DrvOp Of
       'A' : DrvA;
       'B' : DrvC;
     End;
   Until (DrvOp = #27);
   CloseWin(32, 8, 48, 15);
   End;
 If (((GetMediaDrive(0) = 'No instalada') And (GetMediaDrive(1) <> 'No instalada')) And
   ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) = 0))) Then
 Begin
   Repeat
     SetWindow(1,25,80,25);
     SetColors(LightGray, LightGreen); Write(' F4 ');
     SetColors(LightGray, Black); Write('Disco |');
     SetColors(LightGray, Black);
     Write(' Menú de diagnóstico de unidades de disco...          |<');
     SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
     Write('> Salir');
```

```
    SetColors(LightBlue, LightGray);
    CreateWindow(32, 8, 48, 15, 0, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive B ');
    GotoXY(4,3); Write('[b] Drive C ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp in ['A', 'B', #27];
    Case DrvOp Of
     'A' : DrvB;
     'B' : DrvC;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 If (((GetMediaDrive(0) <> 'No instalada') And (GetMediaDrive(1) <> 'No instalada')) And
   ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) = 0))) Then
 Begin
    Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Menú de diagnóstico de unidades de disco...          |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(32, 8, 48, 15, 0, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive A ');
    GotoXY(4,3); Write('[b] Drive B ');
    GotoXY(4,4); Write('[c] Drive C ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp in ['A', 'B', 'C', #27];
    Case DrvOp Of
     'A' : DrvA;
     'B' : DrvB;
     'C' : DrvC;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 If (((GetMediaDrive(0) <> 'No instalada') And (GetMediaDrive(1) <> 'No instalada')) And
   ((GetCapacityDrive(3) <> 0) And (GetCapacityDrive(4) <> 0))) Then
 Begin
    Repeat
    SetWindow(1,25,80,25);
    SetColors(LightGray, LightGreen); Write(' F4 ');
    SetColors(LightGray, Black); Write('Disco |');
    SetColors(LightGray, Black);
    Write(' Menú de diagnóstico de unidades de disco...          |<');
    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
```

```
    CreateWindow(32, 8, 48, 15, 0, 'Drives');
    SetColors(LightBlue, LightGray);
    ClrScr;
    GotoXY(4,2); Write('[a] Drive A ');
    GotoXY(4,3); Write('[b] Drive B ');
    GotoXY(4,4); Write('[c] Drive C ');
    GotoXY(4,5); Write('[d] Drive D ');
    GotoXY(4,6); Write('[Esc] Salir');
    GotoXY(6,8); Write('Opción: ');
    Repeat
      DrvOp := UpCase(ReadKey);
    Until DrvOp in ['A', 'B', 'C', 'D', #27];
    Case DrvOp Of
     'A' : DrvA;
     'B' : DrvB;
     'C' : DrvC;
     'D' : DrvD;
    End;
  Until (DrvOp = #27);
  CloseWin(32, 8, 48, 15);
  End;
 Scrfunc;
End;

(* Tecla de Diagnóstico de Video [F5] *)
 If ((Ch1 = 0) And (Code = 83)) Then
 Begin
   Sonido(800,100);
   Repeat
     SetWindow(1,25,80,25);
     SetColors(LightGray, LightGreen); Write(' F5 ');
     SetColors(LightGray, Black); Write('Video |');
     SetColors(LightGray, Black);
     Write(' Menú de análisis de Video...                    |<');
     SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
     Write('> Salir');
     SetColors(LightBlue, LightGray);
     CreateWindow(29, 10, 53, 15, 0, 'Video');
     SetColors(LightBlue, LightGray);
     ClrScr;
     GotoXY(4,2); Write('[a] Características');
     GotoXY(4,3); Write('[b] Paleta de Color');
     GotoXY(4,4); Write('[Esc] Salir');
     GotoXY(4,6); Write('Opción: ');
     Repeat
       Vdo := UpCase(ReadKey);
     Until Vdo in ['A','B', #27];
     Case Vdo Of
       'A' : Video1;
       'B' : Begin
       If (SI_NO(((EGAVGACtrl Shr $03) And $01)=0]) ='Si' Then
        Begin
          Video2;
        End
       Else
        If (SI_NO[((EGAVGACtrl Shr $03) And $01)=0]) ='No' Then
         Begin
           Sonido(440, 100);
           Delay(100);
           Sonido(440, 100);
           SetWindow(1,25,80,25);
```

```
                SetColors(LightGray, LightGreen); Write(' F5 ');
                SetColors(LightGray, Black); Write('Video ');
                SetColors(LightGray, Black);
                Write(' Paleta de colores...                    ');
                SetColors(Red, LightGray);
                CreateWindow(22, 9, 58, 16, 0, 'Error');
                SetColors(Red, LightGray);
                ClrScr;
                GotoXY(2,2); Write('  No se encontró cargada la Paleta');
                GotoXY(2,3); Write('de Colores, verifique el tipo de ');
                GotoXY(2,4); Write('el tipo de Adaptador instalado en');
                GotoXY(2,5); Write('opción [a] del menú actual.');
                GotoXY(2,7); Write('    Presione ENTER para continuar...');
                EnterKeyPress;
                CloseWin(22, 9, 58, 16);
              End;
            End;
        End;
        Until (Vdo = #27);
        CloseWin(29, 10, 53, 15);
        Scrfunc;
    End;

  (* Tecla de Diagnóstico de Impresoras [F6] *)
  If ((Ch1 = 0) And (Code = 64)) Then
  Begin
    Repeat
      SetWindow(1,25,80,25);
      SetColors(LightGray, LightGreen); Write(' F6 ');
      SetColors(LightGray, Black); Write('Impreso ');
      SetColors(LightGray, Black);
      Write(' Menú de diagnóstico de Impresoras...                  ');
      SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray, Black);
      Write('> Salir');
      Sonido(800, 100);
      SetColors(LightBlue, LightGray);
      CreateWindow(30, 9, 50, 15, 0, 'Impresoras');
      SetColors(LightBlue, LightGray);
      ClrScr;
      GotoXY(6,2); Write('[1] Prueba');
      GotoXY(6,3); Write('[2] Estado');
      GotoXY(6,5); Write('[Esc] Salir');
      GotoXY(7,7); Write('Opción: ');
      Repeat
        Op := UpCase(ReadKey);
      Until Op in ['1','2', #27];
      Case Op Of
        '1' : Begin
                TestPrinter;
              End;
        '2' : Begin
                StateFlagsPrinter;
              End;
    End;
    Until (Op = #27);
    CloseWin(30, 8, 58, 15);
    Scrfunc;
  End;

  (* Tecla de Fin de Programa [F10] *)
  If ((Ch1 = 0) And (Code = 68)) Then
```

```
Begin
  SetWindow(1,25,80,25);
  SetColors(LightGray, LightGreen); Write(' F10 ');
  SetColors(LightGray, Black); Write('Salir |');
  SetColors(LightGray, Black);
  Write(' Finalizar el programa...                    | <');
  SetColors(LightGray, Red); Write('S'); SetColors(LightGray, Black);
  Write('> Salir');
  SetColors(Red, White);
  CreateWindow(64, 22, 76, 22, o, ' Salir ');
  SetColors(LightRed, White);
  ClrScr;
  Write('   (S/N):  ');
  Repeat
    KeyF := Upcase(ReadKey);
  Until KeyF in ['S', 'N'];
  Case KeyF Of
   'N': Begin
          Code := 0;
          CloseWin(64, 22, 76, 22);
          Scrfunc;
        End;
   'S': Code := 68;
  End;
 End;
 Until ((Ch1 = 0) And (Code = 68));
 CloseWin(1, 25, 80, 25);
 SetColors(Black, LightGray);
 ClrScr;
 SetCursorSize(7, 8);
 Sonido(800, 100);
End.
```

V.6 PROGRAMA SIMULA (Simulador de impresora)

El programa SIMULA realiza un trabajo simultáneo con el circuito simulador de impresoras, al entrar al programa se presenta un menú con las siguientes opciones:

- Verificación LEDs:

Se realiza un conteo progresivo de los bits 0 a 7 y retorno (7 a 0) automáticamente; además, se puede realizar la misma prueba manualmente utilizando las teclas [+] y [-], las cuales aumentan o decrementan el conteo (de 0 a 255, código ASCII).

- Verificar banderas:

Al presionar o liberar los switches del circuito simulador se envía información a la computadora; la información recibida nos indica si hay papel, la impresora se encuentra disponible o existe algún tipo de error tal y como una impresora real pudiera manifestar.

- Keyboard LPT:

En esta opción la prueba a realizar es por medio del teclado; al presionar cualquier tecla el código de ésta es enviado hacia el circuito simulador desplegandolo en los LED's.

- Información:

Se presenta la información necesaria para ejecutar adecuadamente este programa.

- Regresar a DOS:

Se da término a la sesión de trabajo, devolviendo el mando a DOS.

El listado de instrucciones correspondiente al programa SIMULA se muestra a continuación:

```
{              Programa :    SimPRN
          Fecha de Inicio :  Enero 18, 1996.
   Fecha de Modificación :   Abril 7, 1996.
}
Uses CRT,
   DOS,
   SysNum,
   Win,
   BIOS;

Type
   SCRSet  = Array[1..1380] of Byte;
    Opcion  = Record
    OpName  : String[20];
    CharSet  : Char;
 End;

   Menu   = Array [1..5] Of Opcion;
Const
   Meses   : Array [1..12] of String[4] = ('Ene', 'Feb', 'Mar', 'Abr',
                                           'Mayo', 'Jun',  'Jul', 'Agst',
                                           'Sept', 'Oct', 'Nov', 'Dic');
   BitOnOff : Array [Boolean] Of Char = (#177, #219);
   TimeSTB  = 500;
Var
   MM              : Word;
   DD              : Word;
   AA              : Word;
   DS              : Word;
   Regs            : Registers;
   Ch1             : Byte;
   Code            : Byte;
   Equipo          : Word Absolute $0040:$0010;
   COM1            : Word Absolute $0040:$0000;
   COM2            : Word Absolute $0040:$0002;
   COM3            : Word Absolute $0040:$0004;
```

```
    COM4            : Word Absolute $0040:$0006;
    LPT1            : Word Absolute $0040:$0008;
    LPT2            : Word Absolute $0040:$000A;
    LPT3            : Word Absolute $0040:$000C;
    LPT4            : Word Absolute $0040:$000E;
    Base            : Word;
    Estado          : Byte;
    Dato            : Byte;
    Control         : Byte;
    Key             : Char;
    X               : Byte;
    Y               : Byte;
    MenuVal         : Menu;
    TMPStr          : String;
    LastItem        : Byte;
    CItem           : Byte;
    SCR             : SCRSet;
    SCRFile         : Text;

Procedure SetShow;
Begin
  SetColors(LightBlue, LightGray);
  CreateWindow(10,  6, 70, 15, S, 'Simulador de Impresoras');
  SetColors(LightBlue, LightGray);
  ClrScr;
  TextColor(LightRed);
  Center('+----------------------------------------------------------+',  1);
  Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+',  2);
  Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+',  3);
  Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+',  4);
  Center('+-+-+-+-+-+-+-+-+-+-----------------------+-+-+-+-+-+-+-+-+',  5);
  Center('+-+-+-+-+-+-+-+-+-+                       +-+-+-+-+-+-+-+-+',  6);
  Center('+-+-+-+-+-+-+-+-+-+-----------------------+-+-+-+-+-+-+-+-+',  7);
  Center('+----------------------------------------------------------+',  8);
  TextColor(LightGray);
  Center('____ _ ___ ___ ____ ____ ____ ___ _',  2);
  Center('___ _ ____ ___ ____ ____ ____ ___',  3);
  Center('___ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _',  4);
  Center('  Versión 1.Beta  ',  6);
  Delay(2000);
  CloseWin(10, 6, 70, 15);
End;

Procedure SetMenuVal(Var MenuSet : Menu);
Begin
  MenuSet[1].OpName  := ' Verificación LEDs  ';
  MenuSet[1].CharSet := 'E';
  MenuSet[2].OpName  := ' Checar banderas    ';
  MenuSet[2].CharSet := 'C';
  MenuSet[3].OpName  := ' KeyBoard LPT       ';
  MenuSet[3].CharSet := 'S';
  MenuSet[4].OpName  := ' Información...     ';
  MenuSet[4].CharSet := 'A';
  MenuSet[5].OpName  := ' Regresar a DOS     ';
  MenuSet[5].CharSet := 'R';
End;

Procedure SetMenuVal1(Var MenuSet : Menu);
Begin
  MenuSet[1].OpName  := ' Puerto LPT1        ';
  MenuSet[1].CharSet := 'P';
```

```
  MenuSet[2].OpName  := ' Puerto LPT2       ';
  MenuSet[2].CharSet := 'P';
  MenuSet[3].OpName  := ' Puerto LPT3       ';
  MenuSet[3].CharSet := 'P';
  MenuSet[4].OpName  := ' Puerto LPT4       ';
  MenuSet[4].CharSet := 'P';
  MenuSet[5].OpName  := ' Regresar...       ';
  MenuSet[5].CharSet := 'R';
End;

Function AddrLPTn(Port : Word) : Word;
Begin
  AddrLPTn := Port;
End;
Procedure InitScr;
Begin
  GetDate(AA, MM, DD, DS);
  SetCursorSize(7, 0);
  ClrScr;
  Window(1, 1, 80, 1);
  SetColors(7, 0);
  ClrScr;
  Write(' SIMPRN V1.Beta ¦    UNAM, Facultad de Ingeniería 1995-96       ');
  Window(1, 25, 80, 25);
  SetColors(7,0);
  ClrScr;
  Write('              ¦ Sistema de Diagnóstico de PC" s e Impresoras ¦ ',
      ' MSDOS', ' ', Lo(DOSVersion), '.', Hi(DOSVersion));
  GotoXY(2, 1); Write(Meses[MM], ' ', DD:2, ', ', AA:4);
  Window(1, 2, 80, 24);
  FillWin(#178, LightGray + Black*15);
End;

Procedure RestoreMenu(Yx : Byte; Item : Byte);
Var
    Y1 : Integer;
Begin
  SetColors(LightBlue, 7);
  CreateWindow(30, 9, 50, 15, s, 'Menu principal');
  SetColors(LightBlue, 7);
  ClrScr;
  For Y1 := 2 to 6 Do
     Begin
       With MenuVal[Y1-1] Do
          Begin
            SetColors(LightBlue, 7);
            GotoXY(2, Y1); Write(OpName:20);
          End;
       SetColors(7, 0);
       GotoXY(1, Yx); Write(MenuVal[ITEM].OpName:21);
     End;
End;

Procedure Sonido(F : Word; T : Word);
Begin
  Sound(F);
  Delay(T);
  NoSound;
End;
Procedure DisplayError;
Begin
```

```
  SetColors(LightRed, LightGray);
  CreateWindow(10, 19, 70, 22, D, 'Error');
  SetColors(LightRed, LightGray);
  ClrScr;
  GotoXY(2, 2); Write('El Adaptador no se encuenta instalado ó bien configurado');
  GotoXY(15, 4); Write('Presione ESC para continuar...');
  ESCKeyPress;
  CloseWin(10, 19, 70, 22);
End;

Procedure SelectPort(Var Base : Word);
Var
  Ky          : Char;
  LItem       : Byte;
  X1          : Byte;
  Y1          : Byte;
  Cl          : Byte;
Begin
  SetMenuVal1(MenuVal);
  RestoreMenu(2, 1);
  LItem := 1;
  Cl   := 1;
  X1   := 1;
  Y1   := 2;
  Cl   := 1;
  RestoreMenu(2, 1);
  SetColors(7, 0);
  GotoXY(1, Y1); Write(MenuVal[Cl].OpName:21);
  Repeat
     Repeat
       Ky := ReadKey;
     Until Ky In [#72, #80, #13, #27];
   Sonido(3750, 2);
   Case Ky Of

       #72 : Begin
             LItem := Cl;
             X1   := Y1;
             Dec(Y1);
             If Y1 <= 1 Then
               Begin
                 Y1 := 6;
                 LItem := 1;
               End;
             Dec(Cl);
             If Cl < 1 Then Cl := 5;
             SetColors(LightBlue, 7);
             GotoXY(1, X1); Write(MenuVal[LItem].OpName:21);
             SetColors(7, 0);
             GotoXY(1, Y1); Write(MenuVal[Cl].OpName:21);
           End;
       #80 : Begin
             LItem := Cl;
             X1   := Y1;
             Inc(Y1);
             If Y1 > 6 Then
               Begin
                 Y1 := 2;
                 LItem := 5;
               End;
             Inc(Cl);
```

```
      If CI > 5 Then CI := 1;
      SetColors(LightBlue, 7);
      GotoXY(1, X1); Write(MenuVal[LItem].OpName:21);
      SetColors(7, 0);
      GotoXY(1, Y1); Write(MenuVal[CI].OpName:21);
    End;
#13 : Begin
      Case CI Of
        1 : Begin
              Sound(50);
              Delay(10 * CI);
              NoSound;
              Base := LPT1;
              IF Base <> 0 Then
                Begin
                  Sonido(4000, 5);
                  Ky := #59;
                End
              Else
                Begin
                  Sonido(100, 100);
                  DisplayError;
                  RestoreMenu(Y1, CI);
                End;
            End;
        2 : Begin
              Sound(75);
              Delay(10 * CI);
              NoSound;
              Base := LPT2;
              IF Base <> 0 Then
                Begin
                  Sonido(4000, 5);
                  Ky := #59;
                End
              Else
                Begin
                  Sonido(100, 100);
                  DisplayError;
                  RestoreMenu(Y1, CI);
                End;
            End;
        3 : Begin
              Sound(100);
              Delay(10 * CI);
              NoSound;
              Base := LPT3;
              IF Base <> 0 Then
                Begin
                  Sonido(4000, 5);
                  Ky := #59;
                End
              Else
                Begin
                  Sonido(100, 100);
                  DisplayError;
                  RestoreMenu(Y1, CI);
                End;
            End;
        4 : Begin
              Sound(75);
```

```
                    Delay(10 * CI);
                    NoSound;
                    Base := LPT4;
                    IF Base <> 0 Then
                      Begin
                        Sonido(4000, 5);
                        Ky := #59;
                      End
                    Else
                      Begin
                        Sonido(100, 100);
                        DisplayError;
                        RestoreMenu(Y1, CI);
                      End;
                  End;
              5 : Begin
                    Sound(50);
                    Delay(10 * CI);
                    NoSound;
                    Base := 0;
                    Ky := #59;
                  End;
            End;
        End;

  Until Ky = #59;
  CloseWin(3, 3, 76, 22);
End;

Function StrBits(StrAux : String): String;
Var
   I   : Integer;
  Str : String;
Begin
  For I := 1 to Length(StrAux) Do
    Begin
      If StrAux[I] = '1' Then
        Str[I] := #219
      Else
        Str[I] := #177;
    End;
    StrBits := Copy(Str, 1, 8);
End;

Procedure PrintBits(PosX, PosY : Integer; Str : String);
Var
  I : Integer;
Begin
  For I := 1 To 8 Do
    Begin
      GotoXY(PosX + I * 2, PosY); Write(Str[I]);
    End;
End;

Procedure PrintBits2(PosX, PosY : Integer; Str : String);
Var
  I : Integer;
Begin
  For I := 1 To 8 Do
    Begin
```

```
      GotoXY(PosX, PosY); Write(Str[i]);
      Inc(PosY);
    End;
End;
Procedure STBLed(X, Y : Byte;  V : Boolean);
Begin
 GotoXY(X, Y);
 If V Then
   Begin
    SetColors(LightBlue, LightRed);
    Write(#178);
   End
 Else
   Begin
    SetColors(LightBlue, LightGreen);
    Write(#219);
   End;
 SetColors(LightBlue, LightGray);
End;

Procedure Enviar;
Var
   Count     : Byte;
   i         : Integer;
   Ch        : Char;
Begin
 SelectPort(Base);
 If Base <> 0 Then
   Begin
    Window(1, 25, 80, 25);
    SetColors(7,0);
    ClrScr;
    Write('          ¦   Envio automático de caracteres a puerto    ¦');
    SetColors(LightGray, LightGreen);
    GotoXY(2, 1); Write('Ver Leds');
    SetColors(LightBlue, LightGray);
    CreateWindow(3, 3, 76, 22, S, 'Enviar caracteres');
    SetColors(LightBlue, LightGray);
    ClrScr;
    FrameItem(3, 2, 17, 2, S, LightGray, 'Datos (Base)');
    FrameItem(20, 2, 31, 2, S, DarkGray, 'Estado');
    FrameItem(34, 2, 45, 2, S, LightGray, 'Control');
    FrameItem(3, 5, 19, 5, S, LightGray, 'Caracter ASCII');
    FrameItem(53, 2, 71, 2, S, LightGray, 'HexaDecimal');
    GotoXY(62, 2); Write('');
    FrameItem(53, 5, 71, 5, S, LightGray, '7 6 5 4 3 2 1 0');
    FrameItem(44, 5, 50, 5, S, LightGray, 'DSTB');
    FrameItem(4,  9, 37, 11, S, LightGray, 'Petición');
    FrameItem(41, 14, 71, 19, S, LightGray, 'Instrucciones');
    FrameItem( 4, 14, 37, 19, S, LightGray, 'Operación');

    (* Manipulación independiente.  No desplegables al inicio *)

    GotoXY( 6, 2); Write(Dec2Hex(Port[Base], 2, 8));
    TextColor(DarkGray);
    GotoXY(22, 2); Write(Dec2Hex(Port[Base+1], 2, 8));
    TextColor(LightGray);
    GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));

    (* Manipulación independiente.  FIN *)
```

```
SetColors(LightBlue, LightRed);
GotoXY(23, 4); Write('Base: ':10, Dec2Hexa(Base, 4), 'h');
GotoXY(23, 5); Write('Estado: ':10, Dec2Hexa(Base+1, 4), 'h');
GotoXY(23, 6); Write('Control: ':10, Dec2Hexa(Base+2, 4), 'h');
SetColors(LightBlue, LightGray);
GotoXY(5, 15); Write('Verificación de estado de LED"s');
Repeat
  Count := 1;
  For I := 0 To 7 Do
     Begin
       Port[Base+0] := Count Shl I;
       GotoXY(6, 2); Write(Dec2Hex(Port[Base], 2, 8));
       GotoXY(11, 5); Write(Chr(Count Shl I));
       GotoXY(58, 2); Write(Copy(Dec2Hexa(Count Shl I, 2), 1, 1));
       GotoXY(66, 2); Write(Copy(Dec2Hexa(Count Shl I, 2), 2, 2));
       SetColors(LightBlue, LightGreen);
       PrintBits(53, 5, StrBits(Dec2Hex(Count Shl I, 2, 8)));
       SetColors(LightBlue, LightGray);
       Port[Base +2] := Port[Base + 2] Xor $01;
       GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
       STBLed(47, 5, (Port[Base+2] And $01)>0);
       Delay(TimeSTB);
       Port[Base +2] := Port[Base + 2] Xor $01;
       GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
       STBLed(47, 5, (Port[Base+2] And $01)>0);
       Delay(300);
     End;
  Count := 128;
  For I := 0 To 7 Do
     Begin
       Port[Base] := Count Shr I;
       GotoXY(6, 2); Write(Dec2Hex(Port[Base], 2, 8));
       GotoXY(11, 5); Write(Chr(Count Shr I));
       GotoXY(58, 2); Write(Copy(Dec2Hexa(Count Shr I, 2), 1, 1));
       GotoXY(66, 2); Write(Copy(Dec2Hexa(Count Shr I, 2), 2, 2));
       SetColors(LightBlue, LightGreen);
       PrintBits(53, 5, StrBits(Dec2Hex(Count Shr I, 2, 8)));
       SetColors(LightBlue, LightGray);
       Port[Base +2] := Port[Base + 2] Xor $01;
       GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
       STBLed(47, 5, (Port[Base+2] And $01)>0);
       Delay(TimeSTB);
       Port[Base +2] := Port[Base + 2] Xor $01;
       GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
       STBLed(47, 5, (Port[Base+2] And $01)>0);
       Delay(300);
     End;
     GotoXY(5, 10); Write('Repetir prueba? S/N:');
     Repeat
       Ch := UpCase(ReadKey);
     Until Ch in ['S', 'N'];
     GotoXY(5, 10); Write(' ':20);
Until Ch = 'N';
CloseWin(3, 3, 76, 22);
Window(1, 25, 80, 25);
SetColors(7,0);
ClrScr;
Write('              ¦     Envio manual de caracteres a puerto      ¦ <');
SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
Write('> Salir');
SetColors(LightGray, LightGreen);
```

```
GotoXY(2, 1); Write('Ver Leds');
SetColors(LightBlue, LightGray);
CreateWindow(3, 3, 76, 22, S, 'Enviar caracteres');
SetColors(LightBlue, LightGray);
ClrScr;
FrameItem(3, 2, 17, 2, S, LightGray, 'Datos [Base]');
FrameItem(20, 2, 31, 2, S, DarkGray, 'Estado');
FrameItem(34, 2, 45, 2, S, LightGray, 'Control');
FrameItem(3, 5, 19, 5, S, LightGray, 'Caracter ASCII');
FrameItem(53, 2, 71, 2, S, LightGray, 'HexaDecimal');
GotoXY(62, 2); Write('|');
FrameItem(53, 5, 71, 5, S, LightGray, '7 6 5 4 3 2 1 0');
FrameItem(44, 5, 50, 5, S, LightGray, 'DSTB');
FrameItem(4,  9, 37, 11, S, LightGray, 'Petición');
FrameItem(41, 14, 71, 19, S, LightGray, 'Instrucciones');
FrameItem( 4, 14, 37, 19, S, LightGray, 'Operación');
GotoXY( 6, 2); Write(Dec2Hex(Port[Base], 2, 8));
TextColor(DarkGray);
GotoXY(22, 2); Write(Dec2Hex(Port[Base+1], 2, 8));
TextColor(LightGray);
GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
SetColors(LightBlue, LightRed);
GotoXY(23, 4); Write('Base: ':10, Dec2Hexa(Base, 4), 'h');
GotoXY(23, 5); Write('Estado: ':10, Dec2Hexa(Base+1, 4), 'h');
GotoXY(23, 6); Write('Control: ':10, Dec2Hexa(Base+2, 4), 'h');
SetColors(LightBlue, LightGray);
GotoXY(5, 10); Write(' ': 20);
ClearBox(5, 15, 35, 15);
GotoXY(5, 15); Write('Escribir en el puerto paralelo');
GotoXY(5, 16); Write('secuencia de bytes de 0-255 y ');
GotoXY(5, 17); Write('viceversa.');
I := 0;
GotoXY(52, 7); Write('[-] Dec      Inc [+]');
Repeat
  Port[Base] := I;
  GotoXY(6, 2); Write(Dec2Hex(Port[Base], 2, 8));
  GotoXY(11, 5); Write(Chr(I));
  GotoXY(58, 2); Write(Copy(Dec2Hexa(I, 2), 1, 1));
  GotoXY(66, 2); Write(Copy(Dec2Hexa(I, 2), 2, 2));
  SetColors(LightBlue, LightGreen);
  PrintBits(53, 5, StrBits(Dec2Hex(I, 2, 8)));
  SetColors(LightBlue, LightGray);
  Port[Base +2] := Port[Base + 2] Xor $01;
  GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
  STBLed(47, 5, (Port[Base+2] And $01)>0);
  Delay(TimeSTB);
  Port[Base +2] := Port[Base + 2] Xor $01;
  GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
  STBLed(47, 5, (Port[Base+2] And $01)>0);
  Delay(TimeSTB);
  FrameItem(41, 14, 71, 19, S, LightGray, 'Instrucciones');
  GotoXY(42, 15); Write('[ - ] Decrementar contador');
  GotoXY(42, 16); Write('[ + ] Incrementar contador');
  GotoXY(42, 17); Write('[ ',Chr(27),' ] Decrementar contador');
  GotoXY(42, 18); Write('[ ',Chr(26),' ] Incrementar contador');
  Repeat
    Ch := ReadKey;
  Until Ch In [#75, #77, #27, #$2B, #$2D];
  Sonido(3750, 2);
  Case Ch Of
    #$2D, #75 : Begin
```

```
                Dec(I);
                If I < 0 Then
                  Begin
                    I := 255;
                  End;
              End;
        #$2B, #77 : Begin
                Inc(I);
                If I > 255 Then
                  Begin
                    I := 0;
                  End;
              End;
      End;
    Until Ch = #27;
    I := 0;
    Count := 0;
    Port[Base] := I;
    GotoXY(6, 2); Write(Dec2Hex(Count, 2, 8));
    GotoXY(11, 5); Write(Chr(Count));
    SetColors(LightBlue, LightGreen);
    PrintBits(53, 5, StrBits(Dec2Hex(Count, 2, 8)));
    SetColors(LightBlue, LightGray);
    GotoXY(58, 2); Write(Copy(Dec2Hex(I, 2), 1, 1));
    GotoXY(66, 2); Write(Copy(Dec2Hex(I, 2), 2, 2));
    ClearBox(41, 14, 70, 19);
    ClearBox( 5, 14, 35, 17);
    CloseWin(3, 3, 76, 22);
  End;
  Port[Base] := 0;
  Port[Base +2] := Port[Base +2] xor $01;
  delay(200);
  Port[Base +2] := Port[Base +2] xor $01;
End;

Procedure Checar;
Begin
 SelectPort(Base);
 If Base <> 0 Then
   Begin
    Window(1, 25, 80, 25);
    SetColors(7,0);
    ClrScr;
    Write('              |  Banderas de Estado con el Simulador  |  <');
    SetColors(LightGray, Red); Write('ENTER'); SetColors(LightGray,Black);
    Write('> Salir');
    SetColors(LightBlue, LightGray);
    CreateWindow(3, 3, 76, 22, 8, 'Checar Banderas de estado');
    SetColors(LightBlue, LightGray);
    ClrScr;
    FrameItem(3, 2, 17, 2, 8, DarkGray, 'Datos [Base]');
    FrameItem(20, 2, 31, 2, 8, LightGray, 'Estado');
    FrameItem(34, 2, 45, 2, 8, DarkGray, 'Control');
    FrameItem(55, 2, 70, 4, 8, LightRed, 'Direcciones');

    (* Manipulación Independiente.  No desplegables al Inicio *)

    SetColors(LightBlue, DarkGray);
    GotoXY( 6, 2); Write(Dec2Hex(Port[Base], 2, 8));
    GotoXY(22, 2); Write(Dec2Hex(Port[Base+1], 2, 8));
    GotoXY(38, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
```

```
    SetColors(LightBlue, LightGray);

    (* Manipulación independiente.  FIN *)

    SetColors(LightBlue, LightRed);
    GotoXY(55, 2); Write('Base: ':10, Dec2Hexa(Base, 4), 'h');
    GotoXY(55, 3); Write('Estado: ':10, Dec2Hexa(Base+1, 4), 'h');
    GotoXY(55, 4); Write('Control: ':10, Dec2Hexa(Base+2, 4), 'h');
    SetColors(LightBlue, LightGray);
    FrameItem( 3, 5, 33, 14, S, LightGray, 'Banderas');
    FrameItem(37, 5, 39, 14, S, LightGray, '');
    GotoXY( 4,  6); Write('No se utiliza':29);
    GotoXY( 4,  7); Write('No se utiliza':29);
    GotoXY( 4,  8); Write('No se utiliza':29);
    GotoXY( 4,  9); Write('Error de impresora':29);
    GotoXY( 4, 10); Write('Impresora seleccionada':29);
    GotoXY( 4, 11); Write('Falta papel':29);
    GotoXY( 4, 12); Write('Reconocimiento de caracteres':29);
    GotoXY( 4, 13); Write('Impresora no ocupada':29);
    GotoXY(43,  9); Write('0 = Error,     1 = No hay error');
    GotoXY(43, 10); Write('0 = No select, 1 = Select');
    GotoXY(43, 11); Write('0 = Tiene,      1 = Falta');
    GotoXY(43, 12); Write('0 = Ack,        1 = Normal');
    GotoXY(43, 13); Write('0 = Ocupada,    1 = No ocupada');
    GotoXY(4, 17); Write(' Oprima los botones del simulador y verificar los cambios de estado');
    Repeat
      {GotoXY( 6, 2); Write(Dec2Hex(Port[Base], 2, 8));}
      GotoXY(22, 2); Write(Dec2Hex(Port[Base+1], 2, 8));
      {GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));}
      GotoXY( 38,  6); Write(BitOnOff[(Port[Base+1] And $01)>0]);
      GotoXY( 38,  7); Write(BitOnOff[((Port[Base+1] Shr $01) And $01)>0] );
      GotoXY( 38,  8); Write(BitOnOff[((Port[Base+1] Shr $02) And $01)>0] );
      GotoXY( 38,  9); Write(BitOnOff[((Port[Base+1] Shr $03) And $01)>0] );
      GotoXY( 38, 10); Write(BitOnOff[((Port[Base+1] Shr $04) And $01)>0] );
      GotoXY( 38, 11); Write(BitOnOff[((Port[Base+1] Shr $05) And $01)>0] );
      GotoXY( 38, 12); Write(BitOnOff[((Port[Base+1] Shr $06) And $01)>0] );
      GotoXY( 38, 13); Write(BitOnOff[((Port[Base+1] Shr $07) And $01)>0] );
    Until KeyPressed;
    GotoXY(25, 19); Write('Presione ENTER para salir...');
    EnterKeyPress;
   End;
  CloseWin(3, 3, 76, 22);
End;
Procedure GetKeystroke(Var Code : Byte;
                       Var Ch1  : Byte);
Var
TMPWord : Word;
Begin
  FillChar(Regs, SizeOf(Regs), 0);
  Regs.AH := $10;
  Intr($16, Regs);
  Code   := Regs.AH;
  Ch1    := Regs.AL;
End;
Procedure KyBrd2LPT;
  Var
    I   : Integer;
Begin
  SelectPort(Base);
  If Base <> 0 Then
    Begin
```

```
Window(1, 25, 80, 25);
SetColors(7,0);
ClrScr;
Write('                ¦ Envío de caracteres al puerto con el teclado ¦ <');
SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
Write('> Salir');
SetColors(LightGray, LightGreen);
GotoXY(2, 1); Write('KeyBoard LPT');
SetColors(LightBtue, LightGray);
CreateWindow(3, 3, 78, 22, S, 'Enviar caracteres');
SetColors(LightBlue, LightGray);
ClrScr;
FrameItem(3, 2, 17, 2, S, LightGray, 'Datos [Base]');
FrameItem(20, 2, 31, 2, S, DarkGray, 'Estado');
FrameItem(34, 2, 45, 2, S, LightGray, 'Control');
FrameItem(3, 5, 19, 5, S, LightGray, 'Caracter ASCII');
FrameItem(53, 2, 71, 2, S, LightGray, 'HexaDecimal');
GotoXY(62, 2); Write('¦');
FrameItem(53, 5, 71, 5, S, LightGray, '7 6 5 4 3 2 1 0');
FrameItem(44, 5, 50, 5, S, LightGray, 'DSTB');
FrameItem(3,  8, 19, 8, S, LightGray, 'Cod. Examina');
FrameItem(41, 14, 71, 19, S, LightGray, 'Instrucciones');
FrameItem( 4, 14, 37, 19, S, LightGray, 'Operación');

(* Manipulación independiente.  No desplegables al inicio *)

GotoXY( 6, 2); Write(Dec2Hex(Port[Base], 2, 8));
TextColor(DarkGray);
GotoXY(22, 2); Write(Dec2Hex(Port[Base+1], 2, 8));
TextColor(LightGray);
GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));

(* Manipulación independiente.  FIN *)

SetColors(LightBlue, LightRed);
GotoXY(23, 4); Write('Base: ':10, Dec2Hexa(Base, 4), 'h');
GotoXY(23, 5); Write('Estado: ':10, Dec2Hexa(Base+1, 4), 'h');
GotoXY(23, 6); Write('Control: ':10, Dec2Hexa(Base+2, 4), 'h');
SetColors(LightBlue, LightGray);
GotoXY(5, 15); Write('Se obtiene digitación del teclado');
GotoXY(5, 16); Write('para transferirla al puerto');
GotoXY(5, 17); Write('paralelo correspondiente.');
GotoXY(42, 15); Write('[ * ] Oprima cualquier tecla');
GotoXY(42, 16); Write('     para enviar el caracter');
GotoXY(42, 17); Write('     al puerto');
Repeat
  GetKeyStroke(Ch1, Code);
      Sonido(3750, 2);
      Port[Base] := Code;
      GotoXY(6, 2); Write(Dec2Hex(Port[Base], 2, 8));
      GotoXY(11, 5); Write(Chr(Code));
      GotoXY(10, 8); Write(Ch1:3);
      GotoXY(58, 2); Write(Copy(Dec2Hexa(Code, 2), 1, 1):1);
      GotoXY(66, 2); Write(Copy(Dec2Hexa(Code, 2), 2, 2):1);
      SetColors(LightBlue, LightGreen);
      PrintBits(53, 5, StrBits(Dec2Hex(Code, 2, 8)));
      SetColors(LightBlue, LightGray);
      Port[Base +2] := Port[Base + 2] Xor $01;
      GotoXY(36, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
      STBLed(47, 5, (Port[Base+2[ And $01)>0);
      Delay(TimeSTB);
```

```
            Port[Base +2] := Port[Base + 2] Xor $01;
            GotoXY(38, 2); Write(Dec2Hex(Port[Base+2], 2, 8));
            STBLed(47, 5, (Port[Base+2] And $01)>0);
            Delay(300);
        Until Code = 27;
        CloseWin(3, 3, 76, 22);
    Port[Base] := 0;
    Port[Base +2] := Port[Base +2] xor $01;
    delay(200);
    Port[Base +2] := Port[Base +2] xor $01;
    End;
End;

Procedure InfoGral;
Var
   AddrPort          : Array [0..15] Of Byte Absolute $0040:0000;
   I                 : Byte;
Procedure PrintAddrPort(X, Y          : Byte;
                        PortNum       : Byte);
Var
  TMPAddr : Word;
  PrePort : String;
Begin
 If (PortNum >= 0) And (PortNum <= 4) Then
   PrePort := 'COM'
 Else
   Preport := 'LPT';
 Case PortNum OF
      1 : TMPAddr := COM1;
      2 : TMPAddr := COM2;
      3 : TMPAddr := COM3;
      4 : TMPAddr := COM4;
      5 : Begin
           PortNum := 1;
           TMPAddr := LPT1;
          End;
      6 : Begin
           PortNum := 2;
           TMPAddr := LPT2;
          End;
      7 : Begin
           PortNum := 3;
           TMPAddr := LPT3;
          End;
      8 : Begin
           PortNum := 4;
           TMPAddr := LPT4;
          End;
    End;
If TMPAddr <> 0 Then
  Begin
   GotoXY(X, Y); Write(PrePort, PortNUM, ': ', Dec2Hexa(TMPAddr, 4), ' h');
  End
Else
  Begin
   GotoXY(X, Y);
   Write(PrePort, PortNum, ': ', '.... h');
  End;
End;
Begin
    Window(1, 25, 80, 25);
```

```
SetColors(7,0);
ClrScr;
Write('              ¦    Información y Ayuda de SIMPRN      ¦ ');
SetColors(LightGray, LightGreen);
GotoXY(2, 1); Write('Información');
MSGWin(20, 23, 59, 23, LightBlue, LightGray, 'Presione ENTER para continuar...');
SetColors(LightBlue, LightGray);
CreateWindow(4, 3, 75, 20, O, 'Información General');
SetColors(LightBlue, LightGray);
ClrScr;
TextColor(LightRed);
Center('+-----------------------------------------------------------+', 5);
Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+', 6);
Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+', 7);
Center('+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+', 6);
Center('+-+-+-+-+-+-+-+-+-+----------------------+-+-+-+-+-+-+-+-+-+', 9);
Center('+-+-+-+-+-+-+-+-+-+                      +-+-+-+-+-+-+-+-+-+', 10);
Center('+-+-+-+-+-+-+-+-+-+----------------------+-+-+-+-+-+-+-+-+-+', 11);
Center('+-----------------------------------------------------------+', 12);
TextColor(LightGray);
Center('____ _ ___ ___ ____ ____ ____ ___ _', 6);
Center('___ _ _ _ _ _ ___ ____ ____ _ _ _', 7);
Center('___ _ _ _ _ _ _ _ _ _ _ ___', 8);
Center('  Versión 1.Beta  ', 10);
GotoXY(34, 14); Write('Diseño y programación:');
GotoXY(34, 16); Write('Victor José Luis Rodríguez Martínez');
GotoXY(34, 17); Write('Francisco Román Ponce Saldaña');
GotoXY(34, 18); Write('Araceli Flores Soto');
EnterKeyPress;
ClrScr;
GotoXY(2,  1); Write('+----------------------+');
GotoXY(2,  2); Write('¦ SIMPRN V1.BETA     ¦');
GotoXY(2,  3); Write('+-----------------------------------------------------------------------');
GotoXY(2,  5); Write('    SIMPRN es un programa diseñado especialmente para manipular los');
GotoXY(2,  6); Write('adaptadores de impresora con que cuenta una computadora. Este ');
GotoXY(2,  7); Write('programa es destinado para realizar pruebas de escritura/lectura de ');
GotoXY(2,  8); Write('los adaptadores de impresión.');
GotoXY(2, 10); Write('    SIMPRN proporciona tres procedimientos de verificación de los');
GotoXY(2, 11); Write('adaptadores de impresión:');
GotoXY(2, 13); Write('                Verificación de estado de leds');
GotoXY(2, 14); Write('                Chequeo de banderas de impresora');
GotoXY(2, 15); Write('                Envío de caracteres con el teclado ');
EnterKeyPress;
ClrScr;
GotoXY(2,  3); Write('    La primera opción sirve para verificar el funcionamiento del cir');
GotoXY(2,  4); Write('cuito de salida basado en el 74LS374 y una regleta de leds.  Este pro');
GotoXY(2,  5); Write('cedimiento consiste en escribir valores en el byte de datos (direc');
GotoXY(2,  6); Write('ción Base), observando el despliegue en pantalla y comparandolo con');
GotoXY(2,  7); Write('el estado que guarda el circuito de salida (74LS245).');
GotoXY(2, 10); Write('    La opción de chequeo de banderas de impresora verifica el estado');
GotoXY(2, 11); Write('que guarda la impresora; las banderas a verificar son:');
GotoXY(2, 13); Write('                Error');
GotoXY(2, 14); Write('                Select');
GotoXY(2, 15); Write('                Paper End');
GotoXY(2, 16); Write('                Acknowledge');
GotoXY(2, 17); Write('                Busy');
EnterKeyPress;
ClrScr;
GotoXY(2,  2); Write('    La tercera opción obtiene la digitación hecha sobre el teclado y');
GotoXY(2,  3); Write('transfiere su código ASCII al registro de datos del adapatador');
GotoXY(2,  4); Write('desplegando tal código en la regleta de leds.');
```

```
    GotoXY(2, 8); Write('  A continuación se proporciona un reporte de la disposición de');
    GotoXY(2, 9); Write('adaptadores de impresora y puertos seriales.');
    EnterKeyPress;
    ClrScr;
    FrameItem(3, 2, 18, 2, S, LightGreen, 'Puertos');
    FrameItem(22, 2, 37, 2, S, LightGreen, 'Puertos');
    GotoXY(4,2); Write('COM"s: ', ((Equipo Shr $09) And $07):3);
    GotoXY(23, 2); Write('LPT"s: ', ((Equipo Shr $0E) And $07):3);
    FrameItem(3, 5, 37, 8, S, LightGreen, 'Direcciones');
    For I := 1 to 4 do
      Begin
        PrintAddrPort(5,i+4, i);
        PrintAddrPort(23,i+4,i+4);
      End;
    FrameItem(42, 2, 71, 16, S, LightRed, 'Observaciones');
    GotoXY(43,2); Write('  El obtener la dirección de');
    GotoXY(43,3); Write('los puertos, seriales y pa--');
    GotoXY(43,4); Write('ralelos, implica consultar');
    GotoXY(43,5); Write('el contenido en la dirección');
    GotoXY(43,6); Write('0040h:0000h; donde respecti-');
    GotoXY(43,7); Write('vamente les corresponden dos');
    GotoXY(43,8); Write('bytes.');
    GotoXY(43,10); Write('  El número de puertos seria');
    GotoXY(43,11); Write('les/paralelos se consulta en');
    GotoXY(43,12); Write('la dirección 0040h:0010h');
    GotoXY(43,14); Write('       Bits 9-11: COM"s');
    GotoXY(43,15); Write('    Bits 14 y 15: LPT"s ');
    FrameItem(3, 11, 37, 12, S, LightGreen, 'BIOS 0040h:0000h');
    GotoXY(5,11);
    Write('COM"s: ');
    For I := 0 to 7 Do
      Begin
        Write(Dec2Hexa(AddrPort[I],2), ' ');
      End;
    GotoXY(5,12);
    Write('LPT"s: ');
    For I := 8 to 15 Do
      Begin
        Write(Dec2Hexa(AddrPort[I],2), ' ');
      End;
    FrameItem(3, 15, 37, 15, S, LightGreen, 'BIOS 0040h:0010h');
    GotoXY(12,15); Write(Dec2Hex(Equipo, 2, 16));
    SetColors(LightBlue, White);
    GotoXY(12, 15); Write((Equipo Shr $F) And $01, (Equipo Shr $E) And $01 );
    GotoXY(16, 15); Write((Equipo Shr $B) And $01,(Equipo Shr $A) And $01,(Equipo Shr $9) And $01);
    SetColors(LightBlue, LightGray);
    EnterKeyPress;
    CloseMSGWin(4, 4, 76, 21);
    CloseWin(10, 22, 70, 22);
  CloseWin(3, 3, 75, 22);
End;

Procedure Ayuda;
Begin
  SetColors(LightBlue, LightGray);
  CreateWindow(6, 3, 72, 22, S, 'Ayuda SIMPRN V1.0');
  SetColors(LightBlue, LightGray);
  ClrScr;
  ReadLn;
  CloseWin(3, 3, 76, 22);
End;
```

```
Begin
  GetDate(AA,MM,DD,Ds);
  CheckBreak := False;
  InitScr;
  SetShow;
  Window(1, 25, 80, 25);
  SetColors(7,0);
  ClrScr;
  Write('              ¦          Simulador de Impresoras          ¦  <');
  SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
  Write('> Salir');
  GotoXY(3, 1); Write(Chr(24), Chr(25), ' Mover');
  SetMenuVal(MenuVal);
  LastItem := 1;
  CItem    := 1;
  X        := 1;
  Y        := 2;
  CItem    := 1;
  RestoreMenu(Y, 1);
  SetColors(7, 0);
  GotoXY(1, Y); Write(MenuVal[CItem].OpName:21);
  Repeat
   Repeat
     Key := ReadKey;
   Until Key In [#72, #80, #13, #27];
   Sonido(3750, 2);
   Case Key Of

      #72 : Begin
            LastItem := CItem;
            X        := Y;
            Dec(Y);
            If Y <= 1 Then
              Begin
                Y := 6;
                LastItem := 1;
              End;
            Dec(CItem);
            If CItem < 1 Then CItem := 5;
            SetColors(LightBlue, 7);
            GotoXY(1, X); Write(MenuVal[LastItem].OpName:21);
            SetColors(7, 0);
            GotoXY(1, Y); Write(MenuVal[CItem].OpName:21);
          End;
      #80 : Begin
            LastItem := CItem;
            X        := Y;
            Inc(Y);
            If Y > 6 Then
              Begin
                Y := 2;
                LastItem := 5;
              End;
            Inc(CItem);
            If CItem > 5 Then CItem := 1;
            SetColors(LightBlue, 7);
            GotoXY(1, X); Write(MenuVal[LastItem].OpName:21);
            SetColors(7, 0);
            GotoXY(1, Y); Write(MenuVal[CItem].OpName:21);
          End;
      #13 : Begin
```

```
Case CItem Of
  1 : Begin
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
      Write('          ¦      Selección de Puertos          ¦ ');
      Write(Chr(24), Chr(25), ' Mover');
      SetColors(LightGray, LightGreen);
      GotoXY(2, 1); Write('Ver Leds');
      Sound(50);
      Delay(10 * CItem);
      NoSound;
      Enviar;
      SetMenuVal(MenuVal);
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
      Write('          ¦      Simulador de Impresoras      ¦ <');
      SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
      Write('> Salir');
      GotoXY(3, 1); Write(Chr(24), Chr(25), ' Mover');
      RestoreMenu(Y, CItem);
    End;
  2 : Begin
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
      Write('          ¦      Selección de puertos          ¦ ');
      Write(Chr(24), Chr(25), ' Mover');
      SetColors(LightGray, LightGreen);
      GotoXY(2, 1); Write('Ver Banderas');
      Sound(75);
      Delay(10 * CItem);
      NoSound;
      Checar;
      SetMenuVal(MenuVal);
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
      Write('          ¦      Simulador de Impresoras      ¦ <');
      SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
      Write('> Salir');
      GotoXY(3, 1); Write(Chr(24), Chr(25), ' Mover');
      RestoreMenu(Y, CItem);
    End;
  3 : Begin
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
      Write('          ¦      Selección de puertos          ¦ ');
      Write(Chr(24), Chr(25), ' Mover');
      SetColors(LightGray, LightGreen);
      GotoXY(2, 1); Write('KeyBoard LPT');
      Sound(100);
      Delay(10 * CItem);
      NoSound;
      KyBrd2LPT;
      SetMenuVal(MenuVal);
      Window(1, 25, 80, 25);
      SetColors(7,0);
      ClrScr;
```

```
                    Write('            ¦        Simulador de Impresoras        ¦ <');
                    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
                    Write('> Salir');
                    GotoXY(3, 1); Write(Chr(24), Chr(25), ' Mover');
                    RestoreMenu(Y, CItem);
                  End;
              4 : Begin
                    Sound(75);
                    Delay(10 * CItem);
                    NoSound;
                    InfoGral;
                    SetMenuVal(MenuVal);
                    Window(1, 25, 80, 25);
                    SetColors(7,0);
                    ClrScr;
                    Write('            ¦        Simulador de Impresoras        ¦ <');
                    SetColors(LightGray, Red); Write('ESC'); SetColors(LightGray,Black);
                    Write('> Salir');
                    GotoXY(3, 1); Write(Chr(24), Chr(25), ' Mover');
                    RestoreMenu(Y, CItem);
                  End;
              5 : Begin
                    Sound(50);
                    Delay(10 * CItem);
                    NoSound;
                    Key := #27;
                  End;
            End;
        End;
    End;
  Until Key = #27;
  Delay(1000);
  CloseWin(20, 9, 50, 15);
  Window(1, 1, 80, 25);
  SetColors(0, 7);
  ClrScr;
  SetCursorSize($7, $8);
  WriteLn('Termina SIMULA...');
End.
```

V.7 UNIDADES UTILIZADAS EN LOS PROGRAMAS

Una unidad es una colección de declaraciones, de procedimientos y funciones que constituyen un programa independiente o módulo. Es el equivalente a las librerías de programas en otros lenguajes. Para poder ejecutarlas es necesario llamarlas desde el programa principal por medio de la declaración *USES*.

Las unidades que se mencionan a continuación, contienen funciones utilizadas en los programas SIMULA y de DIAGNOSTICO:

- DOS:

Es una interfaz entre el programa y las funciones que se deseen realizar (pertenencientes a DOS) sin salir de este.

- CRT:

Ayuda a manejar las funciones desplegadas en pantalla.

- Win:

Con la ayuda de esta unidad se crean las ventanas de presentación, efectos de sombras y marcos con texto.

- **SysNum:**

Realiza las conversiones de bases numéricas (binario, decimal, hexadecimal, etc.) necesarias para la ejecución y presentación de los programas.

- **Comunica:**

Es utilizada para la obtención de las direcciones de los puertos seriales y paralelos.

- **Drives:**

Proporciona información referente a las unidades de disco (flexibles y/o duros), tanto de funcionamiento, capacidad, errores, etc.

- **Equip y Equip2:**

Se realizan operaciones para obtener la información concerniente a la computadora que se este analizando; esta información se refiere a: monitor, mouse, teclado, etc.

- **MseTst:**

Se encarga de realizar la prueba de desempeño del mouse.

- **STDPRN:**

Con esta unidad se lleva a cabo la ejecución de la prueba a impresoras de matriz de puntos en modo EPSON (9 agujas).

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

APENDICE A: FUNCIONES DE DOS Y BIOS

*** ROM BIOS (Read Only Memory Basic Input/Output System)**

El ROM BIOS es un programa que ha sido grabado dentro de una memoria EPROM, el programa es copiado a esta memoria como si fuera copiado a algún disco.

El BIOS es el primer programa que hace uso total de la memoria de la computadora cuando ésta es encendida, ejecutándose a sí mismo sin orden alguna del usuario. Su función es la de checar la existencia y correcto funcionamiento de todos los componentes de la computadora, incluyendo la memoria principal; si el teclado se encuentra conectado; el número de unidades de discos flexibles y discos duros disponibles; inicialización de la impresora y otro tipo de periféricos que se encuentren conectados al sistema. Si todo se encuentra bien, el programa envía una señal al primer disco flexible para ejecutar el programa de arranque del sistema operativo, si no lo encuentra en esta unidad, lo busca en otro disco que esté disponible. Pero si no llega a localizar el sistema operativo, solicita al usuario colocar un disco con el sistema operativo dentro de la primera unidad de discos.

*** Modos de Presentación en Video**

MODO	TIPO	ADAPTADOR	DEFINICION	CAJA	CARACTER	COLORES
00h	texto	CGA[3]	320 x 200	8 x 8	40 x 25	16
		EGA [2,3]	320 x 350	8 x 14	40 x 25	16
		MCGA	320 x 400	8 x 16	40 x 25	16
		VGA [1]	360 x 400	9 x 16	40 x 25	16
01	texto	CGA	320 x 200	8 x 8	40 x 25	16
		EGA [2]	320 x 350	8 x 14	40 x 25	16
		MCGA	320 x 400	8 x 16	40 x 25	16
		VGA [1]	360 x 400	9 x 16	40 x 25	16
02	texto	CGA [3]	640 x 200	8 x 8	80 x 25	16
		EGA [2,3]	640 x 350	8 x 14	80 x 25	16
		MCGA	640 x 400	8 x 16	80 x 25	16
		VGA [1]	720 x 400	9 x 16	80 x 25	16
03	texto	CGA	640 x 200	8 x 8	80 x 25	16
		EGA [2]	640 x 350	8 x 14	80 x 25	16
		MCGA	640 x 400	8 x 16	80 x 25	16
		VGA [1]	720 x 400	9 x 16	80 x 25	16
04	gráfico	CGA/EGA/MCGA/ VGA	320 x 200	8 x 8	40 x 25	4

MODO	TIPO	ADAPTADOR	DEFINICION	CAJA	CARACTER	COLORES
05	gráfico	CGA/EGA [3]	320 x 200	8 x 8	40 x 25	4
		MCGA/VGA	320 x 200	8 x 8	40 x 25	4
06	gráfico	CGA/EGA/MCGA/ VGA	640 x 200	8 x 8	80 x 25	2
07	texto	MDA/EGA	720 x 350	9 x 14	80 x 25	mono
		VGA	720 x 400	9 x 16	80 x 25	
08	gráfico	PCjr	160 x 200	8 x 8	20 x 25	16
09	gráfico	PCjr	320 x 200	8 x 8	40 x 25	16
0A	gráfico	PCjr	640 x 200	8 x 8	80 x 25	4
0B	Reserv.	Reservado				
0C	Reserv.	Reservado				
0D	gráfico	EGA/VGA	320 x 200	8 x 8	40 x 25	16
0E	gráfico	EGA/VGA	640 x 200	8 x 8	80 x 25	16
0F	gráfico	EGA/VGA	640 x 350	8 x 14	80 x 25	mono
10	gráfico	EGA/VGA	640 x 350	8 x 14	80 x 25	16
11	gráfico	MCGA/VGA	640 x 480	8 x 16	80 x 30	2
12	gráfico	VGA	640 x 480	8 x 16	80 x 30	16
13	gráfico	MCGA/VGA	320 x 200	8 x 8	40 x 25	256

[1] Modo VGA ampliado; de otra forma, el VGA puede emular las características ya sea de CGA o de EGA para este modo.

[2] Modo EGA al conectarse a una pantalla de color ampliada; de otra forma, emula las características de CGA para este modo.

[3] Denota tres tonos de gris.

* Palabra de Estado del Equipo.

BIT	DESCRIPCION	OPERACION	VALORES
0	Unidad de disco instalada = 1	((Equipo and 01h) > 0)	True/False
1	Coprocesador matemático instalado = 1	((Equipo Shr 01h) and 01) > 0	True/False
2, 3	RAM del tablero de sistema	((Equipo Shr 02h) and 03h)	00 = 16K 01 = 32K 02 = 48K 03 = 64K
2	Dispositivo apuntador instalado = 1 (PS/2)	((Equipo Shr 02h) and 01h) > 0	True/False
3	No se usa (PS/2)		
4, 5	Modo inicial de video	((Equipo Shr 04h) and 03h)	01 = 40 x 25 color 10 = 80 x 25 color 11 = 80 x 25 mono
6, 7	Número de unidades de disco instaladas	((Equipo Shr 06h) and 03h) + 1	Si el Bit 0 = 1: 00 = 1 unidad 01 = 2 unidades 10 = 3 unidades 11 = 4 unidades
8	No se usa	Ninguna	Ninguno
9, 0A, 0B	Número de tarjetas de puerto serial	((Equipo Shr 09h) and 07H	
0C	Adaptador de juegos instalado = 1	((Equipo Shr 0Ch) and 01h > 0	True/False
0D	Modem interno instalado = 1 (PS/2)	((Equipo Shr 0Dh) and 01h) > 0	True/False
0E, 0F	Número de adaptadores de impresora	((Equipo Shr 0Eh) and 01h) >0	

Equipo = apuntador a la tabla de BIOS, en la dirección 0040h:0013h
Shr = Shift Right (corrimiento a la derecha)
h = Notación utilizada para valores hexadecimales

* Bits de Estado del Controlador de Disco

BIT = 1	VALOR	DESCRIPCION	OPERACION
0	01h	Orden inválida para el controlador.	((Drive and 01h) > 0)
1	02h	Marca de dirección no encontrada.	((Drive Shr 01h) and 01h) > 0
0,1	03h	Disco protegido contra escritura.	((Drive and 01h) > 0) and (((Drive Shr 01h and 01h) > 0)
2	04h	Sector solicitado no encontrado.	((Drive Shr 02h) and 01h > 0)
1, 2	06h	Línea de cambio de disco activa.	(((Drive Shr 01h) and 01h) > 0) and (((Drive Shr 02h) and 01h) > 0)
3	08h	Desbordamiento de capacidad de DMA.	((Drive Shr 03h) and 01h) > 0
0,3	09h	Intento de DMA con frontera 64K	((Drive and 01h) > 0) and (((Drive Shr 03h and 01h) > 0)
2, 3	0Ch	Medios inválidos.	(((Drive Shr 02h) and 01h) > 0) and (((Drive Shr 03h and 01h) > 0)
4	10h	Error de CRC (Check Redundancy Cyclic).	((Drive Shr 04) and 01h) > 0
5	20h	Error del controlador.	((Drive Shr 05) and 01h) > 0
6	40h	Error de búsqueda.	((Drive Shr 06) and 01h) > 0
7	80h	Tiempo perdido de disco (Drive no Redy).	((Drive Shr 07) and 01h) > 0

Nota: La variable drive apunta a la localidad de memoria RAM, donde se localiza el registro de control de disco. Aunque se puede utilizar el valor numérico del registro de estado del controlador de disco, es preferible realizar las operaciones booleanas correspondientes para un mejor control del programa al utilizar sólo una variable de referencia.

*** Bits de Estado de Puerto**

BIT = 1	DESCRIPCION	OPERACION
0	Datos preparados.	(COMx and 01h) > 0
1	Error de desbordamiento de capacidad.	((COMx Shr 01h) and 01h) > 0
2	Error de paridad.	((COMx Shr 02h) and 01h) > 0
3	Error de formulación.	((COMx Shr 03h) and 01h) > 0
4	Interrupción detectada.	((COMx Shr 04h) and 01h) > 0
5	Registro de detención para transmisión vacía (THR).	((COMx Shr 05h) and 01h) > 0
6	Registro de desplazamiento para transmisión (TSR).	((COMx Shr 06h) and 01h) > 0
7	Tiempo de espera.	((COMx Shr 07h) and 01h) > 0

Nota: Esta tabla es utilizada para verificar el estado de cualquiera de los puertos seriales después de cada una de las operaciones (Init, Write, Read, Etc). En este caso, la variable COMx tipo Byte apunta al registro de estado del puerto en uso.

* Bits de Estado de Modem.

BIT = 1	DESCRIPCION	OPERACION
0	Cambio de estado de clear to send (CTS).	(Modem and 01h) > 0
1	Cambio de estado de data set ready (DSR).	((Modem Shr 01h) and 01h) > 0
2	Indicador de llamada de extremo final.	((Modem Shr 02h) and 01h) > 0
3	Cambio de señal de linea de recepción.	((Modem Shr 03h) and 01h) > 0
4	Clear to Send (CTS) (despejado para enviar).	((Modem Shr 04h) and 01h) > 0
5	Data Set Ready (DSR) (conjunto de datos preparado).	((Modem Shr 05h) and 01h) > 0
6	Indicador de llamada.	((Modem Shr 06h) and 01h) > 0
7	Señal de linea de recepción detectada.	((Modem Shr 07h) and 01h) > 0

*** Códigos de Retorno de Servicios de Cassette**

CODIGO	DESCRIPCION
00h	Orden inválida.
01h	Error de CRC.
02h	Pérdida de transiciones de datos.
03h	Ningún dato localizado en cinta.
04h	Datos no encontrados (sólo para PCjr).
86h	Ningún puerto de cassette disponible.

*** Tabla de Descriptor Global (GDT)**

DESPLAZAMIENTO	DESCRIPCION
00h	Ficticio (vale cero).
08h	Localidad de segmento de datos de GDT (vale cero).
10h	Apuntador de GDT fuente.
18h	Apuntador de GDT destino.
20h	Apuntador al segmento de código de BIOS, con valor inicial de cero. BIOS usará esta área para crear el segmento de código de modo protegido.
28h	Apuntador al segmento de pila de BIOS, con valor inicial de cero. BIOS usará esta área para crear el segmento de pila de modo protegido.

* Disposición de GDT Fuente/Destino

DESPLAZAMIENTO	DESCRIPCION
00h	Límite de segmento.
02h	Dirección física de segmento de 24 bits.
05h	Derechos de acceso a datos (vale 93h).
06h	Palabra reservada (debe ser cero).

*** Bits de Estado de Impresión.**

BIT	DESCRIPCION	OPERACION
0	Tiempo de espera.	((Printer and 01h) > 0)
1, 2	No se utiliza.	
3	Error de Entrada/Salida.	((Printer Shr 03h) and 01h) > 0
4	Impresora seleccionada.	((Printer Shr 04h) and 01h) > 0
5	Falta papel.	((Printer Shr 05h) and 01h) > 0
6	Reconocido.	((Printer Shr 06h) and 01h) > 0
7	Impresora desocupada	((Printer Shr 07h) and 01h) > 0

*** FUNCIONES DE BIOS**

BIOS significa *Basic Input/OutPut System*. Sus interrupciones y servicios se describen a continuación:

INT 10h
Servicio 00h
Set Video Mode (Fijar modo de video)

Llamada: AH := 00h
AL := Modo de representación

Devuelve: Nada

Comentario: Fija el modo de video despejando la pantalla. Si desea conservar el contenido de la pantalla en sistemas EGA, MCGA y VGA, fije el bit 7 de AL en 1.

```
AL := AL XOr 80h
AL := AL + ModeVideo
```

INT 10h
Servicio 01h
Set cursor type (Fija tipo de cursor)

Llamada: AH := 01h
CH := Línea inicial de examinación (límite superior) en bits 0-4
CL := Línea final de examinación (límite inferior) en bits 0-4

Devuelve: Nada

Comentario: Para modos monocromáticos, la línea de examinación inicial por omisión es 0Bh, y la final es 0Ch. Para modo de color, la línea de examinación inicial por omisión es 06h, y la de terminación es 07h.

INT 10h
Servicio 02h
Set cursor position (Fija posición del cursor)

Llamada: AH := 02h
BH := Número de página (0 para los modo gráficos)
DH := Renglón
DL := Columna

Devuelve: Nada

Comentario:

Páginas	Modos	Adaptadores
0-7	00h, 01h	CGA, EGA, MCGA, VGA
0-3	02h, 03h	CGA
0-7	02h, 03h	EGA, MCGA, VGA
0	07h	MDA
0-7	07h	EGA, VGA

INT 10h
Servicio 03h
Read Cursor position and configuration (Leer posición y configuración del cursor)

Llamada: AH := 03h
BH := Número de página

Devuelve: BH := Número de página de video
CH := Línea de examinación inicial para el cursor
CL := Línea de examinación final para el cursor
DH := Renglón
DL := Columna

Comentario: Devuelve las líneas de examinación inicial y final de cursor, y su posición actual.

INT 10h
Servicio 05h
Read cursor position and configuration (Leer posición y configuración del cursor)

Llamada: AH := 05h
AL := Número de página seleccionada

Devuelve: Nada

Comentario:

Páginas	Modos	Adaptadores
0-7	00h, 01h	CGA, EGA, MCGA, VGA
0-3	02h, 03h	CGA
0-7	02h, 03h	EGA, MCGA, VGA
0-3	07h, 0Dh	EGA, VGA
0-3	0Eh	EGA, VGA
01	OFh, 10h	EGA, VGA

INT 10h
Servicio 06h
Scroll Windows Up (Desplazar la ventana hacia arriba)

Llamada: AH := 06h
AL := Número de renglones por desplazar
BH := Atributo usado para área en blanco
CH := Renglón, esquina superior izquierda
CL := Columna, esquina superior izquierda
DH := Renglón, esquina inferior derecha
DL := Columna, esquina inferior derecha

Devuelve: Nada

Comentario: Despeja una ventana con el atributo específico o desplaza la ventana hacia arriba determinado número de renglones. Todos los renglones de la ventana se desplazan hacia arriba y se añaden renglones en blanco en la parte inferior. Para despejar una ventana, se fija AL en cero o en un valor mayor que el número de renglones de la ventana.

INT 10h
Servicio 07h
Scroll Windows Down (Desplazar la ventana hacia abajo)

Llamada: AH := 07h
AL := Número de renglones por desplazar

BH := Atributo usado para área en blanco
CH := Renglón, esquina superior izquierda
CL := Columna, esquina superior izquierda
DH := Renglón, esquina inferior derecha
DL := Columna, esquina inferior derecha

Devuelve: Nada

Comentario: Despeja una ventana con el atributo específico, o desplaza la ventana hacia abajo determinado número de renglones. Todos los renglones de la ventana se desplazan hacia abajo y se añaden renglones en blanco en la parte superior. Para despejar una ventana, se fija AL en cero o en un valor mayor que el número de renglones de la ventana.

INT 10h
Servicio 08h
Read Character and attribute (Leer caracter y atributo)

Llamada: AH := 08h
BH := Página de presentación visual

Devuelve: AH := Byte de atributo
AL := Carácter ASCII

Comentario: Obtiene el código del caracter y atributo desplegado en la posición actual del cursor.

INT 10h
Servicio 09h
Write Character and attribute (Escribir caracter y atributo)

Llamada: AH := 09h
AL := Caracter ASCII
BH := Página de presentación visual
BL := Byte de atributo del caracter en AL
CX := Número de caracteres por escribir

Devuelve: Nada

Comentario: Escribe el caracter seleccionado el número de veces deseado. Esta función escribe hasta 65536 caracteres en modo texto.

INT 10h
Servicio 0Ch
Write Graphics Pixel (Escribir pixel para gráficos)

Llamada: AH := 0Ch

AL := Valor del color
BH := Número de página
CX := Número de columna del pixel
DX := Número de renglón del pixel

Devuelve: Nada

Comentarios: Ninguno

INT 10h
Servicio 0Dh
Read Graphics Pixel (Leer pixel para gráficos)

Llamada: AH := 0Dh
BH := Número de página
CX := Número de columna del pixel
DX := Número de renglón del pixel

Devuelve: AL := Valor del color

Comentarios: Ninguno

INT 10h
Servicio 0Fh
Get current display mode (Obtener modo de presentación actual)

Llamada: AH := 0Fh

Devuelve: AH := Número de columnas en pantalla
AL := Modo de presentación
BH := Página activa de presentación

Comentarios: Ninguno

INT 10h
Servicio 13h
Write string (Escribir cadena)

Llamada: AH := 13h
AL := Modo de escritura
BH := Página de video
BL := Atributo (modos de escritura 0 y 1)
CX := Longitud de la cadena
DH := Renglón en el cual se escribirá la cadena
DL := Columna en la cual se escribirá la cadena
ES := Segmento de la cadena

BP := Offset de la cadena

Devuelve: Nada

Comentarios:

Modo	Comentario
0	Atributo BL. Cadena sólo de caracteres, no actualiza posición del cursor.
1	Atributo BL. Cadena sólo de caracteres, actualiza la posición del cursor.
2	Atributo BL. Cadena alterna caracter/atributo, no actualiza posición del cursor.
3	Atributo BL. Cadena alterno caracter/atributo, actualiza posición del cursor.

INT 11h
Servicio
Get equipment status (Obtener estado del equipo)

Llamada: Ninguna

Devuelve: AX := Estado del equipo

Comentarios: Regresa la configuración del equipo (Coprocesador, Video, Puertos, Drives, etc). Consulte la tabla de estado de equipo.

INT 12h
Servicio
Get memory size (Obtener tamaño de memoria base)

Llamada: Ninguna

Devuelve: AX := Número de bloques de memoria de 1K

Comentarios: Regresa el número de bloques de 1024 bytes instalados como memoria. Para obtener el tamaño exácto aplique la siguiente fórmula:

MemoriaBase := (NumBloques x 1024) Bytes

INT 13h
Servicio 00h
Reset floppy disk system (Restablecer sistema de disco flexible)

Llamada: AH := 00h
DL := Número de unidad (basado en cero)

Bit 7 = 0 para disco flexible, 1 para disco duro

Devuelve: Bandera de acarreo en cero si hubo éxito
Bandera de acarreo en uno si hubo error
AH := Código de retorno

Comentarios: Forza la unidad a tirar de las cabezas hacia la pista cero.

INT 13h
Servicio 01h
Get floppy disk system status (Obtener estado del sistema de disco flexible)

Llamada: AH := 01h

Devuelve: AH := Byte de estado

Comentarios: Con esta función se puede comprobar el estado que guarda el controlador de unidades de disco flexible después de cada operación.

INT 13h
Servicio 02h
Read floppy disk (Leer disco flexible)

Llamada:
AH := 02h
AL := Número de sectores por transferir (1-9)
ES:BX := Apuntador al buffer de disco del usuario
CH := Número de pista
CL := Número de sector
DH := Número de cabeza
DL := Número de unidad (0 a 3)

Devuelve:
Bandera de acarreo en cero si hubo éxito
Bandera de acarreo en uno si hubo error
AL := Número de sectores transferidos
AH := Byte de estado

Comentarios: Transfiere uno o más sectores del disco a la memoria principal.

INT 13h
Servicio 03h
Write disk sectors (Escribir sectores del disco)

Llamada:
AH := 03h
AL := Número de sectores por transferir
ES:BX := Apuntador al buffer de disco del usuario
CH := Número de pista

CL := Número de sector
DH := Número de cabeza
DL := Número de unidad (0 a 3)

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
AL := Número de sectores transferidos
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Escribe en uno o más sectores del disco la información almacenada en la memoria principal.

INT 13h
Servicio 04h
Verify disk sectors (verificar sectores de disco)

Llamada: AH := 03h
AL := Número de sectores por transferir
CH := Número de pista
CL := Número de sector
DH := Número de cabeza
DL := Número de unidad (0 a 3)

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Verifica el CRC (Cyclic Redundancy Check) de los sectores del disco.

INT 13h
Servicio 08h
Return disk drive parameters (Devolver parámetros de unidad de disco)

Llamada: AH := 08h
DL := Número de unidad (basado en 0)

Devuelve: Bandera de acarreo en cero si hubo éxito
CH := Número de pistas por lado
CL := Número de sectores por pista
DH := Número de lados
DL := Número de unidades consecutivas conectadas
ES:DI := Apuntador a tabla de parámetros de disco de 11 bytes
BL := Valor de tipo de unidad válido de CMOS

01h = 5¼, 360Kb, 40 pistas
02h = 5¼, 1.2Mb, 80 pistas

03h = 3½, 720Kb, 80 pistas
05h = 3½, 1.44Mb, 80 pistas

Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Permite verificar las características del drive. Proporciona acceso a la tabla de parámetros del drive.

Offset	Significado
00h	Primer byte especificado.
01h	Segundo byte especificado.
02h	Número de "tic-tacs" de reloj antes de apagar el motor de la unidad.
03h	Número de bytes por sector: 00h = 128, 01h = 256, 02h= 512, 03h = 1024
04h	Sectores por pista.
05h	Longitud de marca de terminación.
06h	Longitud de datos.
07h	Longitud de marca de terminación para formato.
08h	Byte de relleno para formato.
09h	Tiempo de colocación de la cabeza en milisegundos.
0Ah	Tiempo de arranque del motor en octavos de segundo.

INT 13h
Servicio 09h
Initialize fixed disk table (Asignar valores iniciales a la tabla de disco duro)

Llamada: AH := 09h
DL := Número de unidad de disco duro

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Los números de las unidades de disco duro inician con el valor 80h para el primer disco, 81h para el segundo, etc. La información es obtenida de la tabla de parámetros del disco (Vectores de interrupción 41h, 42h).

INT 13h
Servicio 0Ah
Read long sector (Leer sector largo)

Llamada: AH := 0Ah
AL := Número de sectores (1 a 121)

ES := Segmento de buffer de datos
BX := Desplazamiento en el segmento de buffer de datos
CH := Pista (0 a 1023)
CL := Sector (1 a 17)
DH := Número de cabeza (de 0 a 15)
DL := Número de unidad (80h, 81h, etc.)

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Ninguno

INT 13h
Servicio 0Bh
Write long sector (Escribir sector largo)

Llamada: AH := 0Bh
AL := Número de sectores (1 a 121)
ES := Segmento de buffer de datos
BX := Desplazamiento en el segmento de buffer de datos
CH := Pista (0 a 1023)
CL := Sector (1 a 17)
DH := Número de cabeza (de 0 a 15)
DL := Número de unidad (80h, 81h, etc.)

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Ninguno

INT 13h
Servicio 0Ch
Seek Cylinder (Buscar Cilindro)

Llamada: AH := 0Ch
CH := Pista de orden inferior
CL := Pista de orden alto
DH := Número de cabeza
DL := Número de unidad de disco duro

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0

Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Ninguno

INT 13h
Servicio 0Dh
Alternate disk reset (Restablecer el disco)

Llamada: AH := 0Dh
DL := Número de unidad de disco duro

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := 0
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Disponible en PC AT. Esta función es idéntica a la 13h/00h.

INT 13h
Servicio 15h
Return DASD (Direct Access Storage Device) Type
Devolver tipo de dispositivo de almacenamiento de acceso directo.

Llamada: AH := 15h
DL := Número de unidad

Devuelve: Bandera de acarreo en cero si hubo éxito
AH := Tipo de unidad de DASD
00 La unidad requerida no esta disponible
01 Unidad presente, no se puede detectar cambio de disco
02 Unidad presente, se puede detectar cambio de disco
03 Disco duro
CX := Segmento de número de sectores de disco duro
DX := Desplazamiento en el segmento de número de sectores de disco duro
Bandera de acarreo en uno si hubo error
AH := Byte de estado

Comentarios: Ninguno

INT 13h
Servicio 16h
Read disk change line Status (Leer estado de línea de cambio de disco)

Llamada: AH := 16h

DL := Número de unidad

Devuelve: Bandera de acarreo en cero
AH := 0 Disco no cambiado
Bandera de acarreo en uno
AH := 0 Error
6 Disco cambiado

Comentarios: Ninguno

INT 14h
Servicio 00h
Initialize communications port (Preparar puerto de comunicaciones)

Llamada: AH := 00h
AL := Parámetro de inicialización
DX := Número de puerto PC/AT
0 COM1
1 COM2
2 COM3
3 COM4

Devuelve: AH := Estado de puerto
AL := Estado de modem

Comentarios:

Bits 7, 6, 5	Bits 4, 3	Bit 2	Bits 1, 0
Velocidad en bauds	Paridad	Bits de alto	Longitud de palabra
000 = 110 001 = 150 010 = 300 011 = 600 100 = 1200 101 = 2400 110 = 4800 111 = 9600	X0 = ninguna 01 = impar 10 = Ninguna 11 = par	0 = Un bit 1 = Dos bits	10 = Siete bits 11 = Ocho bits

INT 14h
Servicio 01h
Write character to communications port (Escribir caracter al puerto de comunicaciones)

Llamada: AH := 01h
AL := Código de examinación del caracter
DX := Número del puerto

Devuelve: AH := Bit 7 = 0 (no hubo error)
AH := Bit 7 = 1 (error); los bits 0 a 6 muestran la causa de la falla

Comentarios: Antes de utilizar esta función debe inicializarse el puerto

INT 14h
Servicio 02h
Read character to communications port (Leer caracter del puerto de comunicaciones)

Llamada: AH := 01h
DX := Número del puerto

Devuelve: AH := Bit 7 = 0 (no hubo error)
AL := Código de examinación del caracter
AH := Bit 7 = 1 (error); los bits 0 a 6 muestran la causa de la falla

Comentarios: Antes de utilizar esta función debe inicializarse el puerto

INT 14h
Servicio 03h
Request communications port status (Solicitar el estado del puerto de comunicaciones)

Llamada: AH := 03h
DX := Número de puerto

Devuelve: AH := Estado del puerto
AL := Estado del modem

Comentarios: Ninguno

INT 14h
Servicio 04h
Extended initialization (PS/2) (Preparación ampliada (PS/2))

Llamada: AH := 04h
AL := Valor de interrupción
BH := Paridad
BL := Bits de alto
CH := Longitud de datos
CL := Velocidad de transmisión (bps)
DX := Número de puerto

Devuelve: AH := Estado del puerto
AL := Estado del modem

Comentarios:

Registro	Significado	Disposiciones	Significado
AL	Interrupción	00h	No hay interrupción
		01h	Interrupción
BH	Paridad	00h	No hay paridad
		01h	Paridad impar
		02h	Paridad par
		03h	Paridad de adhesión impar
		04h	Paridad de adhesión par
BL	Bit de alto	00h	Un bit de alto
		01h	Dos bits de alto (1.5, CH = 00h)
CH	Longitud de datos	00h	Cinco bits
		01h	Seis bits
		02h	Siete bits
		03h	Ocho bits
CL	Velocidad	00h	110 bps
		01h	150 bps
		02h	300 bps
		03h	600 bps
		04h	1200 bps
		05h	2400 bps
		06h	4800 bps
		07h	9600 bps
		08h	19200 bps

INT 14h
Servicio 05h
Extended communications port control (PS/2) (Control del puerto de comunicaciones ampliado (PS/2))

Llamada: **AH := 05h**
AL := Leer o escribir en el registro de control de modem
00h = Leer
01h = Escribir
BL := Registro de control de modem (si AL := 01h véase tabla en comentarios)
DX := Número de puerto

Devuelve: **AH := Estado del puerto**
AL := Estado del modem
BL := Registro de control de modem

Comentarios: **Permite la lectura/escritura en el registro de control de modem asociado con el** puerto RS232 deseado. **Los bits de BL se definen en la siguiente tabla:**

Bit = 1	Descripción
0	Terminal de datos preparada (DTR)
1	Solicitud de envio (RST)
2	Fuera uno (Out 1)
3	Fuera dos (Out 2)
4	Prueba de retroalimentación
5, 6, 7	Reservado

INT 15h
Servicio C0h

Return system configuration parameters (Devolver parámetros de configuración del sistema)

Llamada: AH := C0h

Devuelve: PC, Pc*jr*: Bandera de acarreo en uno, AH := 80h
PC XT BIOS (11/08/82), PC/AT BIOS (1/10/84):
Bandera de acarreo en uno, AH := 86h
Todo los demás:
ES:BX Apuntador a la tabla de descriptor del sistema en ROM

Comentarios: La tabla de descriptor de sistema en ROM contiene información útil sobre el sistema. La siguiente tabla muestra el significado de las entradas:

Desplazamiento	Significado
00h	Conteo de bytes de los datos subsecuentes (mínimo 8)
02h	Byte modelo
03h	Byte submodelo
04h	Nivel de revisión de BIOS (00h = primera liberación)
05h	Información de características: Bit 0 = X Reservado Bit 1 = 0 Canal de E/S de línea de PC Bit 1 = 1 Arquitectura de microcanal Bit 2 = 1 Asignado a EBDA Bit 3 = 1 Espera para evento externo se permite Bit 4 = 1 Intercepción de teclado por INT 09h Bit 5 = 1 Reloj de tiempo real presente Bit 6 = 1 Segundo microcircuito de interrupción presente Bit 7 = 1 Canal DMA 3 usado por BIOS de disco duro.
06h - 09h	Reservado

El byte modelo contenido en el desplazamiento 02h de la tabla de descriptor del sistema debería ser igual al byte de identificación del sistema (almacenado en la dirección F000:FFFE). El byte submodelo (desplazamiento 03h) puede usarse para una identificación adicional del sistema.

INT 16h
Servicio 00h
Read keyboard character (Leer caracter de teclado)

Llamada: AH := 00h

Devuelve: AH := Código de examinación de teclado
AL := Código de caracter ASCII

Comentarios: Espera y lee un sólo caracter del buffer del teclado y lo devuelve junto con su código de examinación. El buffer del teclado por lo general se localiza en 0040:001A.

INT 16h
Servicio 01h
Read keyboard status (Leer estado del teclado)

Llamada: AH := 01h

Devuelve: Bandera de acarreo en cero (se oprimió una tecla)
AH := Código de examinación
AL := Caracter ASCII
Bandera de acarreo en uno (no se oprimió tecla alguna)

Comentarios: Ninguno

INT 16h
Servicio 02h
Return keyboard flags (Devolver banderas de teclado)

Llamada: AH := 02h

Devuelve: AL := Byte de banderas de teclado de ROM BIOS

Comentarios: Devuelve el estado de los conmutadores biestables y de las teclas Shift del registro de estado de BIOS mantenido en la localidad de memoria 0040:0017. La siguiente tabla muestra el significado de los bits de registro AL regresando de la función:

Bit 0 = 1	Tecla Shift derecha oprimida
Bit 1 = 1	Tecla Shift izquierda oprimida
Bit 2 = 1	Tecla Ctrl oprimida
Bit 3 = 1	Tecla Alt oprimida

Bit 4 = 1 Scroll Lock habilitado
Bit 5 = 1 Num Lock habilitado
Bit 6 = 1 Caps Lock habilitado
Bit 7 = 1 Tecla Insert activada

INT 16h
Servicio 10h
Get keystroke (Obtener digitación)
Llamada: AH := 10h

Devuelve: AH := Código de examinación
AL := Caracter ASCII

Comentarios: Sólo trabaja en los sistemas PC/AT y PS/2 con teclados expandidos. Permite el reconocimiento de teclas similares.

INT 16h
Servicio 11h
Check keyboard (Verificar teclado)

Llamada: AH := 11h

Devuelve: AH := Código de examinación
AL := Caracter ASCII
Bandera de acarreo en uno si no hay alguna digitación disponible

Comentarios: Ninguno

INT 16h
Servicio 12h
Get keyboard status flags (Obtener banderas de estado del teclado)

Llamada: AH := 12h

Devuelve: AL := Bandera de estado uno
AH := Bandera de estado dos

Comentarios: Esta función sólo trabaja en los sistemas PC/AT y PS/2 con teclados expandidos, es como la INT 16h/02h, añadiendo la segunda bandera de estado del teclado.

Bandera de estado uno:

Bit 0 = 1 Tecla Shift derecha oprimida
Bit 1 = 1 Tecla Shift izquierda oprimida
Bit 2 = 1 Tecla Ctrl oprimida

Bit 3 = 1	Tecla Alt oprimida
Bit 4 = 1	Scroll Lock habilitado
Bit 5 = 1	Num Lock habilitado
Bit 6 = 1	Caps Lock habilitado
Bit 7 = 1	Tecla Insert activada

Bandera de estado dos:

Bit 0 = 1	Tecla Ctrl izquierda oprimida
Bit 1 = 1	Tecla Alt izquierda oprimida
Bit 2 = 1	Tecla Ctrl derecha oprimida
Bit 3 = 1	Tecla Alt derecha oprimida
Bit 4 = 1	Tecla Scroll Lock oprimida
Bit 5 = 1	Num Lock oprimida
Bit 6 = 1	Caps Lock oprimida
Bit 7 = 1	Tecla SysReq oprimida

INT 17h
Servicio 00h
Write character to printer (Escribir caracter en impresora)

Llamada: AH := 00h
AL := Caracter
DX := Número de impresora (0 a 2)

Devuelve: AH := Estado de impresora

Comentarios: Escribe el caracter especificado en el puerto de impresora y devuelve el estado actual de la impresora.

INT 17h
Servicio 01h
Initialize printer port (Preparar puerto de impresión)

Llamada: AH := 01h
DX := Número de impresora (0 a 2)

Devuelve: AH := Estado de la impresora

Comentarios: Prepara el puerto paralelo de la impresora y retorna el estado de éste.
La función dirige la sucesión de caracteres 08h y 0Ch al puerto de impresora.

INT 17h
Servicio 02h
Request printer port status (Solicitar estado de puerto de impresión)

Llamada: AH := 02h
DX := Número de impresora (0 a 2)

Devuelve: AH := Estado de la impresora

Comentarios: Devuelve el estado del puerto de impresora paralelo especificado. Si está utilizando un PC/AT, PC/XT286 o PS/2 y BIOS determina que la impresora esta ocupada, BIOS ejecutará una INT 15h/90h (dispositivo ocupado [DEVICE BUSY])

*** FUNCIONES DE DOS**

Los servicios ofrecidos por DOS cubren las interrupciones 20h a 2Fh y se describen a continuación:

INT 21h
Servicio 05h
Printer output (Salida de imprasora)

Llamada: AH := 05h
DL := Datos de 8 bits para imprimir STDPRN (Unidad diseñada en Pascal para el Programa de Diagnóstico)

Devuelve: Nada

Comentarios: Espera hasta que la impresora está lista y luego envía un byte. La detección de Ctrl C y Ctrl Break durante esta función ocasiona la ejecución de la INT 23h (Vector de interrupción).

INT 21h
Servicio 0Eh
Select Disk (Seleccionar disco)

Llamada: AH := 0Bh
DL := Número de unidad (A = 0 a Z = 25)

Devuelve: AL := Ultimo número de unidad (A = 1 a Z = 26)

Comentarios: Ninguno

INT 21h
Servicio 19h
Get default drive (Obtener unidad por omisión)

Llamada: AH := 19h

Devuelve: AL := Número de unidad actual (A = 0 a Z = 25)

Comentarios: Ninguno

INT 21h
Servicio 1Bh
Get allocation table information (Obtener información de la tabla de asignación)

Llamada: AH := 1Bh

Devuelve: AL := Sectores por cluster
CX := Bytes por sector físico
DX := Cluster por disco
DS := Segmento del byte descriptor de medio de almacenamiento
BX := Desplazamiento sobre el segmento del byte descriptor de medios de almacenamiento

Comentarios: Válido para versiones superiores a 2. DS:BX apunta al byte descriptor de medios contenido en la FAT (Tabla de asignación de archivos), pero en la versión 1 apunta a la FAT en memoria. El byte descriptor de medios o FAT ID puede usarse para identificar el formato de los medios de almacenamiento de acuerdo a la siguiente información:

F0h	No identificable/18 sectores, 80 pistas
F8h	Disco duro
F9h	De 2 lados, 15 sectores por pista (1.2 MB)
F9h	De 2 lados, 9 sectores por pista (720 KB)
FCh	De 1 sólo lado 9 sectores por pista
FDh	De 2 lados, 9 sectores por pista (360 KB)
FEh	De 1 sólo lado, 8 sectores por pista
FFh	De 2 lados, 8 sectores por pista

INT 21h
Servicio 1Ch
Get allocation table information for specific drive (Obtener información de la tabla de asignación para drive específico)

Llamada: AH := 1Ch
DL := Número de unidad (Unidad actual = 0, A = 1 a Z = 26)

Devuelve: AL := Sectores por cluster
CX := Bytes por sector físico
DX := Cluster por disco
DS := Segmento del byte descriptor de medio de almacenamiento
BX := Desplazamiento sobre el segmento del byte descriptor de medios de almacenamiento

Comentarios: Válido para versiones superiores a 2. DS:BX apunta al byte descriptor de medios contenido en la FAT (Tabla de asignación de archivos), pero en la versión 1 apunta a la FAT en memoria. El byte descriptor de medios o FAT ID puede usarse para identificar el formato de los medios de almacenamiento de acuerdo con la siguiente información:

F0h	No identificable/18 sectores, 80 pistas
F8h	Disco duro
F9h	De 2 lados, 15 sectores por pista (1.2 MB)
F9h	De 2 lados, 9 sectores por pista (720 KB)
FCh	De 1 sólo lado 9 sectores por pista
FDh	De 2 lados, 9 sectores por pista (360 KB)
FEh	De 1 sólo lado, 8 sectores por pista
FFh	De 2 lados, 8 sectores por pista

INT 33h
Servicio 00h
Reset Mouse and get status (Reiniciar mouse y obtener estado)

Llamada: AX := 0000h

Devuelve: Si el soporte para mouse esta disponible:
AX := FFFFh
BX := Número de botones
Si no está disponible:
AX := 0000h

Comentarios: Después de realizar la llamada a este procedimiento el controlador del mouse es inicializado:

- El apuntador del mouse aparece en el centro de la pantalla.
- La página de video se fija en cero.
- Se fijan los valores ya predeterminados para los tipos de apuntadores del mouse.
- El controlador de eventos del usuario es deshabilitado.
- Los valores del mouse (1/200 inch) horizontal se fijan en 8x8, y el vertical en 16x8.

INT 33h
Servicio 01h
Show mouse pointer (Presentar apuntador del mouse)

Llamada: AX := 0001h

Devuelve: Nada

Comentarios: Ninguno

INT 33h
Servicio 02h
Hide mouse pointer (Ocultar apuntador del mouse)

Llamada: AX := 0002h

Devuelve: Nada

Comentarios: Ninguno

INT 33h
Servicio 03h
Get mouse position and button status (Obtener posición del mouse y estado de los botones)

Llamada: AX := 0003h

Devuelve: BX := Estado de los botones
Significado de Bits:
0 Botón izquierdo presionado
1 Botón derecho presionado
2 Botón central presionado
3-15 Reservados
CX := Coordenada horizontal (X)
DX := Coordenada vertical (Y)

Comentarios: Los valores de las coordenadas son dados de acuerdo con el modo de despliegue actual. Posición superior izquierda (X,Y) = (0,0).

INT 33h
Servicio 07h
Set horizontal limits for pointer (Fijar limites horizontales para el apuntador)

Llamada: AX := 0007h
CX := Coordenada horizontal (X) mínima
DX := Coordenada horizontal (X) máxima

Devuelve: Nada

Comentarios: Si el valor mínimo (CX) es más grande que el valor máximo (DX), los dos valores son intercambiados entre sí.

INT 33h
Servicio 08h
Set vertical limits for pointer (Fijar limites verticales para el apuntador)

Llamada: AX := 0008h
CX := Coordenada Vertical (Y) mínima
DX := Coordenada Vertical (Y) máxima

Comentarios: Si el valor mínimo (CX) es más grande que el valor máximo (DX), los dos valores son intercambiados entre si.

INT 33h
Servicio 10h
Set mouse pointer exclusion area (Fijar área de exclusión para el apuntador del mouse)

Llamada: AX := 0010h
CX := Coordenada (X) superior izquierda
DX := Coordenada (Y) superior izquierda
S I := Coordenada (X) inferior derecha
D I := Coordenada (Y) inferior derecha

Devuelve: Nada

Comentarios: El área de exclusión puede ser modificada llamando nuevamente esta función o utilizando la función 00h ó 01h.

INT 33h
Servicio 1Fh
Disable mouse driver (Deshabilitar controlador de mouse)

Llamada: AX := 001Fh

Devuelve: Si se realizó la función
AX := 001Fh
ES := Segmento del controlador
BX := Desplazamiento sobre el segmento

Si no se realizó
AX := FFFFh

Comentarios: Ninguno

INT 33h
Servicio 20h
Enable mouse driver (Habilitar controlador de mouse)

Llamada: AX := 0020h

Devuelve: Nada

Comentarios: Ninguno

INT 33h
Servicio 22h
Set lenguage for mouse driver messages (Fijar el lenguaje para los mensajes del controlador)

Llamada: AX := 0022h
BX := Número del lenguaje

0	Inglés
1	Francés
2	Holandés
3	Alemán
4	Sueco
5	Predeterminado
6	Español
7	Portugués
8	Italiano

Devuelve: Nada

Comentario: Debe verificarse la presencia del controlador del mouse. Esta función se encuentra disponible sólo en las versiones internacionales de Microsoft Mouse.

INT 33h
Servicio 23h
Get language for mouse driver messages (Obtener el lenguaje para los mensajes del controlador)

Llamada: AX := 0022h

Devuelve: BX := Número del lenguaje

0	Inglés
1	Francés
2	Holandés
3	Alemán
4	Sueco
5	Predeterminado
6	Español
7	Portugués
8	Italiano

Comentario: Debe verificarse la presencia del controlador del mouse. Esta función se encuentra disponible sólo en las versiones internacionales de Microsoft Mouse.

INT 33h
Servicio 24h
Get mouse information (Obtener información del mouse)

Llamada: AX := 0024h

Devuelve: BX := Versión del controlador
BL := Relación de versión
CH := Tipo de mouse:
1 Bus mouse
2 Serial mouse
3 InPort mouse
4 PS/2 mouse
5 HP mouse
CL := Número de IRQ
0 PS/2
2, 3, 4, 5, ó 7 Número de IRQ

Comentarios: Ninguno

*** Area de datos BIOS**

El BIOS utiliza el sistema RAM para almacenar los datos de especificación. El segmento asignado para este propósito es el 0040h. La tabla que se presenta a continuación se extrae de "IBM PC CMOS BIOS", sin realizar la traducción de la misma, para evitar confusión de términos.

Desplazamiento	Servicio BIOS	Descripción
00h	INT 14h	I/O Address of up to 4 serial communications adapters
08h	INT 17h	I/O Address of up to 4 parallel printer adapters
10h	INT 11h	Number of devices installed, Where:
		Bit 15-14 = Number of printer adapters
		Bit 13-12 = Reserved
		Bit 11-9 = Number of asynchronous adapters
		Bit 8 = Reserved
		Bit 7-6 = Number of disk drives, where
		00b = 1 Disk drive
		01b = 2 Disk drives
		Bit 5-4 = Initial video mode, where:
		00b = EGA/VGA or PGA
		01b = 40x25 Color
		10b = 80x25 Color
		11b = 80x25 Black and White
		Bit 3 = Reserved
		Bit 2 = Pointing device
		Bit 1 = 1 if math Coprocessor
		Bit 0 = Diskette available for boot

Desplazamiento	Servicio BIOS	Descripción
12h	POST	Manufacturing test port (AT only)
13h	INT 12h	Installed memory in Kilobytes
15h	POST	Manufacturing test port (AT only)
17h	INT 16h	Keyboard shift flags, where: Bit 7 = 1 Insert active Bit 6 = 1 Caps Lock active Bit 5 = 1 Num Lock active Bit 4 = 1 Scroll Lock active Bit 3 = 1 Alt pressed Bit 2 = 1 Ctrl pressed Bit 1 = 1 Left Shift pressed Bit 0 = 1 Right Shift pressed
18h	INT 16h	Extended Keyboard Shift flags, where: Bit 7 = 1 Insert pressed Bit 6 = 1 Caps Lock pressed Bit 5 = 1 Num Lock pressed Bit 4 = 1 Scroll Lock pressed Bit 3 = 1 Ctrl-Num Lock state active Bit 2 = 1 Sys Req pressed Bit 1 = 1 Left Shift pressed Bit 0 = 1 Right Shift pressed
19h	INT 16h	Work area for alt key an numeric keypad input
1Ah	INT 16h	Pointer to next character in keyboard buffer
1Ch	INT 16h	Pointer to first available sport in keyboard buffer
1Eh	INT 16h	Keyboard buffer of 16 word entries
3Eh	INT 13h	Diskette drive recalibrate status, where: Bit 7 = 1 Diskette hardware interrupt has occurred Bit 6-4 = No used Bit 3-2 = Reserved Bit 1 = 1 recalibrate drive 1 Bit 0 = 1 recalibrate drive 0
3Fh	INT 13h	Diskette drive motor status, Where: Bit 7 = 1 current operation is a write or format 0 current operation is a read or verify Bit 6 = Reserved Bit 5-4 = Drive select states where: 00b = Drive 0 selected 01b = Drive 1 selected 10b = Reserved 11b = Reserved Bit 1 = Drive 1 motor is On Bit 0 = Drive 0 motor is On
40h	INT 13h	Diskette motor time-out count

Desplazamiento	Servicio BIOS	Descripción
41h	INT 13h	Diskette status return code, where: Bit 7 = 1 Drive no ready Bit 6 = 1 Seek error occurred Bit 5 = 1 Diskette controller failed Bit 4-0 = Error codes, where: 00h = No error 01h = Illegal function was requested 02h = Address mark not found 03h = Write protect error 04h = Sector not found 06h = Drive door was opened 08h = DMA overrun error 09h = DMA Boundary error 0Ch = Media type unknown 10h = CRC failed on disk read
42h-48h	INT 13h	Diskette controller status bytes and command butes for fixed disk controller
49h	INT 10h	Video mode setting
4Ah	INT 10h	Number of colums on screen
4Ch	INT 10h	Current page size (video)
4Eh	INT 10h	Current page address (video)
50h	INT 10h	Cursor position as 6845. Two bytes/page. First byte of each pair is column; secon byte is row. 0,0 is upper left corner of screen.
60h	INT 10h	Cursor type defined as 6845 video chip-compatible startin an ending scan lines. High byte is startin scan line; low is ending scan line.
62h	INT 10h	Current page number (video)
63h	INT 10h	6845-compatible I/O port number for current mode (Port 03D4h or 03B4) (video)
65h	INT 10h	Current mode select register
66h	INT 10h	Current palette value
67h-6Ah	POST	Pointer to resent code upon system reset for real mode re-entry.
6Bh	POST	Last unexpected interrupt that occurred.
6Ch	INT 1Ah	Timer count. Number of ticks since midnight (Four bytes)
70h	INT 1Ah	24 hour roll over flag
72h	POST	System reset flag (AT only), where: 1234h = Bypass memory test (also warm boot) 4321h = Preserve memory 0064h = Burn in mode

Desplazamiento	Servicio BIOS	Descripción
74h	INT 13h	Status from last fixed disk operation, where: 00h = No error 01h = Invalid function request 02h = Address mark not found 03h = Write protect error 04h = Sector not found 05h = Reset failed 07h = Drive parameter activity failed 08h = DMA overrun on operation 09h = DMA boundary error 0Ah = Bad sector flag detected 0Bh = Bad track detected 0Dh = Invalid number of sectors on format 0Eh = Control data address mark detected 0Fh = DMA arbitration level out of range. 10h = Uncorrectable ECC o CRC error 11h = ECC corrected data error 20h = General controller failure 40h = Seek operation failed 80h = Time-Out AAh = Drive not ready BBh = Undefined error occurred CCh = Write fault on seleected drive E0h = Status error/error register is 0 FFh = Sense operation failed
75h	INT 13h	Number of fixed disk
76h	INT 13h	Fixed disk control byte
77h	INT 17h	Fixed disk por offset
78h	INT 14h	Printer time-out table (ports 0-3)
7Ch	INT 16h	Serial time-out table (ports 0-3)
80h	INT 16h	Offset to start of keyboard buffer (From segment 40h)
82h	INT 10h	Offset to en of keyboard buffer (from segment 40h)
84h	INT 10h	Number of rows on screen (24/25) (VGA only)
85h	INT 10h	Character height (bytes/character) (VGA only)
87h	INT 10h	Video control bits (VGA only), where: Bit 7 = Clear RAM Bit 6-5 = Memory on adapter as follows: 00b = 64K 01b = 128K 10b = 192K 11b = 256K Bit 4 = No used Bit 3 = 0 EGA/VGA-compatible adapter active Bit 2 = Wait for display enable Bit 1 = 0 Color or ECD monitor is attached to EGA/VGA-compatible adapter 1 Monochrome monitor is attached to EGA/VGA-compatible adapter

Desplazamiento	Servicio BIOS	Descripción
		Bit 0 = 0 Translate cursor video modes 0-3 when using ECD monitor in 350 line mode 1 Inhibit cursor translation
88h	INT 10h	EGA/VGA switch data (VGA only), Where: Bit 7-4 = Feature connector bits 3-0, respectively Bit 3-0 = Option switches 3-0, respectively
89h	INT 10h	EGA/VGA control bits (VGA only), where: Bit 7 = 200 lines Bit 6-5 = Reserved Bit 4 = 400 lines Bit 3 = No palete load Bit 2 = Mono monitor Bit 1 = Gray scalling Bit 0 = Reserved
8Ah	INT 13h	Index into DCC table (VGA only)
8Bh	INT 13h	Diskette data rate information Bit 7-6 = Last data rate set by controller, where: 00b = 500 Kilobits/second (Kbs) 01b = 300 Kbs 10b = 250 Kbs Bit 5-4 = Last diskette drive step rate selected Bit 3-2 = Data transfer rate at operation start. Where: 00b = 500 Kbs 01b = 300 Kbs 10b = 250 Kbs Bit 1-0 = Reserved
8Ch	INT 13h	Fixed disk status register (AT only)
8Dh	INT 13h	Fixed disk error register (AT only)
8Eh	INT 13h	Fixed disk interrrupt flag (AT only)
8Fh	INT 13h	Diskette controller information, where: Bit 7 = Reserved Bit 6 = 1 Drive determined for drive 1 Bit 5 = 1 Drive 1 is multirate Bit 4 = 1 Drive 1 supports change line Bit 3 = Reserved Bit 2 = 1 Drive determined for drive 0 Bit 1 = 1 Drive 0 is multirate Bit 0 = 1 Drive 0 supports change line
90h-91h	INT 13h	Diskette controller information. Where: (one byte per drive, drive 0 at 40:90h; drive 1 at 40:91h) Bit 7-6 = Data transfer rate, where: 00b = 500 Kbs 01b = 300 Kbs 10b = 250 Kbs Bit 5 = 1 Double stepping required (360Kb media/1.2Mb drive)

Desplazamiento	Servicio BIOS	Descripción
		Bit 4 = 1 Known media in drive Bit 3 = Reserved Bit 2-0 = Definitions on return to user: 111b = 720K media in 720K or 1.44Mb drive; or 1.44Mb media in 1.44Mb drive 101b = Known 1.2Mb media in 1.2Mb drive 100b = Known 360K media in 1.2Mb drive 011b = Known 360K media in 360K drive 010b = Trying 1.2Mb media in 1.2Mb drive 001b = Trying 360K media in 1.2Mb drive 000b = Trying 360K media in 360K drive
92h	INT 13h	Diskette device service work area. Each entry is first diskette device service value tried. One byte per drive. Drive 0 at 92h, drive 1 at 93h.
94h	INT 13h	Current track number for both drives. One byte per drive. Drive 0 at 94h, drive 1 at 93h.
96h	INT 16h	Kerboard status byte: Bit 7 = 1 Read ID in progress Bit 6 = 1 Last code was first ID Bit 5 = 1 Forced Num Lock Bit 4 = 1 101/102 keyboard used Bit 3 = 1 Right Alt active Bit 2 = 1 Right Ctrl active Bit 1 = 1 Last code was E0h Bit 0 = 1 Las code was E1h
97h	INT 16h	Keyboard LED status byte: Bit 7 = Error flag for keyboard command Bit 6 = LED UpDate in progress Bit 5 = Resend received from keyboard Bit 4 = ACK received from keyboard Bit 3 = Reserved Bit 2 = Current status of Caps Lock LED Bit 1 = Current status of Num Lock LED Bit 0 = Current status of Scroll Lock LED
98h	INT 15h	User wait flag offset address (AT only)
9Ah	INT 15h	User wait flag segment address (AT only)
9Ch	INT 15h	Least significant byte of wait count (AT only)
9Eh	INT 15h	Most significant byte of wait count (AT only)
A0h	INT 15h	Wait active flag. (AT only) Bit 7 = 1 Wait time elapsed Bit 6-1 = Reserved Bit 0 = 1 INT 15h, AH = 86h occurred

Desplazamiento	Servicio BIOS	Descripción
A8h	INT 10h	Pointer video parameters and overrides (in segment:offset format) VGA only.
B0h-B5h		Reserved
B6h-B8h	POST	Reserved for POST (At only)
C0h-CDh		Reserved
CEh	INT 1Ah	Count days since 1-1-80
CFh-FFh		Reserved
100h	INT 05h	Print screen status byte.

APENDICE B: FUENTE DE PODER Y SECUENCIA DE INICIO DE DOS

La fuente de poder o alimentación utilizada en las PC's es del tipo conmutada, la cual requiere de un mínimo de carga para tener una operación adecuada. En condiciones normales, no ocurre ningún daño si la fuente es alimentada con voltaje de AC si no se tiene conectada alguna carga en cualquiera de sus conectores de salida. En tal caso , la fuente automáticamente se apaga.

La fuente de poder, provee a la PC de una señal de OK (power good) indicando la operación adecuada de la fuente. Cuando la fuente se encuentra apagada y se conmuta a encendida, transcurre un lapso mínimo de 1 s y la señal de OK es generada, asumiendo que no hubo problemas. Esta señal, es una señal lógica AND proveniente de la salida de voltaje de DC y de la entrada de AC percibidas en la señal. La señal de OK, también es compatible con TTL, teniendo nivel alto en operaciones normales y nivel bajo en condiciones de falla. La señal de falla de AC causa que la señal de OK entre a un nivel bajo de por lo menos 1 ms antes de cualquier salida de voltaje de falla por debajo de los límites de regulación. El punto de operación como medida de referencia de 1 ms es de operación normal con un mínimo de voltaje de línea y un máximo de carga. La señal de salida del voltaje de DC percibida mantiene la señal de OK en un nivel bajo cuando la fuente es encendida hasta que todos los voltajes de salida hayan alcanzado su mínimo nivel.

Caracteristicas de una fuente de poder con problemas:

- No trabaja el ventilador.
- La computadora no enciende.
- Olor a quemado, sonidos de click (posiblemente un capacitor dañado).
- Beep continuo.
- Series de más de 2 beeps
- Mensaje de error 02xx

En el caso de los mensajes de error y de beeps, puede variar en cada PC, dependiendo del BIOS que contenga cada una.

Los problemas con la fuente pueden ser causados por:

- El switch de elección de voltaje que se encuentra en la parte posterior de la fuente no ha sido fijado correctamente, este debe seleccionarse dependiendo del voltaje de línea que se tenga.
- El cable de alimentación se encuentra defectuoso. Ocasionalmente los cables de alimentación no se encuentran en lugares adecuados y la gente cuando camina cerca de la PC llega a pisar y dañar los cables, por lo que se debe cuidar donde se colocan los cables.
- Otro tipo de daño en el cable, son las puntas que no realizan buen contacto con el enchufe, esto debido a mal estado por oxidación, se encuentran dobladas o no se encuentran unidas al cable.
- La fuente de poder puede no presentar voltaje alguno de salida, esto posiblemente a que se quemó un fusible, si este es el problema, se debe de reemplazar el fusible dañado por otro equivalente.
- Si se tiene voltaje de salida en la fuente, el ventilador de la fuente trabaja, el monitor enciende, pero este no muestra nada en la pantalla, el problema pueden ser los contactos de la fuente que

no se encuentran conectados en la tarjeta madre. Se debe de revisar adecuadamente las conexiones de la fuente, puesto que puede no tenerse un buen contacto y causar que la alimentación de voltaje sea intermitente.

- Por el continuo uso el switch de encendido y apagado, al accionarse puede quedarse estancado, siempre encendido o siempre apagado, por lo que este debe ser sustituido por otro en buen estado.

- Cuando la computadora da inicio pero nada aparece en la pantalla se emiten beeps continuos; la luz indicadora del disco duro se encuentra intermitente y no completa su proceso de arranque, da inicio nuevamente el arranque o a la mitad de la ejecución de un programa el sistema realiza un arranque en caliente por si mismo; hay insuficiencia de potencia de salida de la fuente, por lo que es necesario cambiarla por una fuente de mayor potencia (watts).

- En algunas ocasiones es demasiada la corriente que se demanda a la fuente por tarjetas de expansión, por lo que puede apagar completamente el sistema. Se puede realizar el cambio de la fuente de poder o remover alguna de las tarjetas de expansión.

- Cuando en la línea de alimentación existen fluctuaciones de voltaje, esto puede ser causado por tormentas eléctricas, apagando y encendiendo aparatos eléctricos conectados en un mismo contacto. Las fluctuaciones pueden ser en forma de pequeños rizos, los cuales pueden causar un sobrecalentamiento en los componentes de los circuitos. Una alimentación de AC con fluctuaciones, puede causar continuos errores de paridad inconsistentemente en locaciones de memoria. Muchas de estas variaciones pueden ser causadas por la unidad de fuente de poder. Para evitar tales problemas se debe de tener una fuente de poder de alta calidad para tener un mejor suministro de energía y un buen desempeño de la PC.

Para evitar tener fluctuaciones de voltaje de AC se debe tener cuidado en los siguientes aspectos:

a) No se debe de conectar en el mismo contacto a la computadora con un circuito de maquinaria pesada, aires acondicionados, herramientas, televisiones, impresoras láser, máquinas copiadoras, cafeteras eléctricas, aspiradoras o cualquier otro aparato que se encienda o apague frecuentemente. Preferiblemente la computadora debe de conectarse en su propio contacto.

b) La computadora debe de conectarse a tierra utilizando las tres puntas del enchufe.

c) Se debe de evitar utilizar extensiones de cable de alimentación cuando sea posible, puesto que el ruido en la línea eléctrica se puede incrementar. Si se utiliza una extensión, se debe conocer el calibre del cable a utilizar debido a la longitud que se necesite para manejar adecuadamente la carga.

- Para tratar de eliminar en lo más mínimo los rizos en la línea de voltaje de AC, se debe de utilizar el protector adecuado, ya sean MOV's (Metal Oxide Varistor), acondicionadores o reguladores de voltaje, filtros, fuentes ininterrumpibles, etc.

Secuencia de inicio de DOS

Si se llega a tener algún problema con el sistema de la computadora durante su inicio al ser encendida, se puede determinar el problema ocurrido siguiendo una serie de eventos que son ejecutados en secuencia. Si se conocen los eventos que deben ocurrir, podremos encontrar la causa del problema. A continuación se enumeran los eventos típicos que ocurren en la secuencia de inicio de DOS:

1. Se suministra energía eléctrica al sistema.

2. La proporción de energía da inicio a una autoprueba. Cuando todos los voltajes y niveles de corriente son adecuados, la fuente indica que la energía se encuentra estable y manda una señal de OK a la tarjeta madre. El tiempo que transcurre entre el encendido y la verificación de estabilidad de energía, normalmente se encuentra entre 0.1 y 0.5 segundos.

3. Si el reloj del microprocesador no recibe la señal de OK, causa que el reloj se detenga y genere una señal de reset (reinicio) para el microprocesador.

4. El microprocesador comienza a ejecutar el código de ROM BIOS comenzando en la dirección de memoria FFFF:0000. Esta localidad de solamente 16 bytes que se encuentra al final del espacio disponible en ROM, contiene una instrucción JMP (salto) para la dirección actual de inicio del ROM BIOS.

5. El ROM BIOS ejecuta una prueba al hardware central para verificar la funcionabilidad de los sistemas básicos. Cualquier error que llegue a ocurrir es indicado por medio de un código de audio (un beep), esto debido a que el sistema de video no ha sido inicializado todavía.

6. El BIOS ejecuta una búsqueda en el video ROM entre las localidades de memoria C000:0000 a C780:0000, buscando el programa de video adaptador de ROM BIOS contenido en una tarjeta adaptadora de video conectada en un *slot*. Si el ROM BIOS de video es encontrado, una prueba de verificación es ejecutada. Si pasa la prueba entonces la ROM es ejecutada y el código de video en ROM inicializa el video adaptador desplegando un cursor en la pantalla. Si en la prueba se obtiene una falla aparece el siguiente mensaje:

C000 ROM Error

7. Si BIOS no encuentra el ROM video adaptador, este utiliza el driver de video de la tarjeta madre para dar inicio al despliegue de video y un cursor aparece en la pantalla.

8. El ROM BIOS de la tarjeta madre localiza la memoria ROM a través de las localidades C800:0000 a la DF80:0000 en incrementos de 2K para cualquier ROM localizada en otra tarjeta adaptadora. Si es encontrada otra ROM se le realiza una prueba y es ejecutada. Cualquier otra ROM en las tarjetas adaptadoras pueden alterar las rutinas existentes de BIOS, así como establecer nuevas.

9. Cualquier falla en la prueba de diagnóstico de cualquier módulo de ROM causa que el siguiente mensaje aparezca:

XXXX ROM Error

10. La dirección XXXX indica el segmento de dirección de la falla en el módulo ROM.

11. El ROM BIOS revisa el valor de la palabra en la localidad de memoria 0000:0472 para ver si el inicio del sistema es en caliente o frío. Si el valor de la palabra es de 1234h en esta localidad, es una bandera que indica un inicio en caliente, lo que causa que la porción de prueba de memoria del POST (Power On Self Test, Autoprueba de energía) no se realize. Cualquier otro valor de palabra en esta localidad indica que se ha realizado el inicio en frío y es realizado completo el POST.

12. Si se indica inicio en frio se ejecuta el POST, cualquier error encontrado durante el POST es reportada por medio de una combinación de audio y video desplegando mensajes de error. Si el POST ha tenido éxito es indicado por medio de un beep.

13. El ROM BIOS busca el sector de arranque que contenga el DOS en el cilindro 0, cabeza 0 y sector 1 del drive A. Este sector es cargado (alojado) en la memoria en la dirección 0000:7C00 y es verificado. Si un disco se encuentra en el drive pero el sector no puede ser leído ó si el disco no se encuntra presente, el BIOS continua con el siguiente paso.

14. Si el primer byte del DOS en el sector de arranque que es cargado del disco en el drive A es menor, mayor o igual que 06h y las primeras nueve palabras contienen el mismo modelo de datos, se despliega el siguiente mensaje de error y el sistema se detiene.

602 Diskette Boot Record Error

15. Si el sector de arranque no puede ser leido del drive A, el BIOS busca una partición maestra del sector de arranque en el cilindro 0, sector 1 del primer disco que se encuentre preparado con el sector de arranque. Si este sector es encontrado, se carga en memoria en la dirección 0000:7C00 y busca una *"firma"* (los dos últimos bytes de la partición maestra).

16. Si la firma del sector de arranque no es igual a 55AAh es invocada la interrupción 18h (Int 18h), esto ocurre en la mayor parte de los sistemas. En un sistema IBM PS/2 es deplegado un mensaje con características gráficas especiales representando que se debe de introducir un disco en el drive A y presionar la tecla F1. Para ningún sistema PS/2 hecho por IBM, una Int 18h ejecutará el ROM BIOS basado en el intérprete BASIC, en cualquier otro sistema IBM compatible se despliega un mensaje indicando que ocurrió algún tipo de error de arranque. Por ejemplo, un sistema con ROM BIOS para AT de Phoenix despliega el siguiente mensaje:

No Boot Device Available
Strike F1 to Retry Boot, F2 for Setup Utility

17. El programa de la partición maestra del sector de arranque busca su tabla de partición para una entrada con un sistema indicador de byte; si el programa encuentra tal entrada, se carga la partición extendida del sector de arranque en la localidad indicada, la cual cuenta con una tabla adicional en la que se busca otra partición extendida. Si otra partición es encontrada, es cargada al sector de arranque de la localidad indicada y la búsqueda continua hasta que no existan más particiones extendidas o se alcance el número máximo de particiones que es de 24.

18. La partición maestra del sector de arranque busca su tabla de partición por medio de un byte indicador de arranque marcando una partición activa.

19. En un sistema IBM si ninguna de las particiones es marcada activa (bootable), el ROM BIOS basado en BASIC es invocado. En muchos sistemas compatibles con IBM algunos tipos de mensajes de error de disco son desplegados.

20. Si cualquier indicador de arranque en la partición maestra grabado en la tabla es inválido o si más de uno indica una partición activa, aparece el siguiente mensaje y el sistema se detiene:

Invalid partition table

21. Si una partición activa es encontrada en la partición maestra del sector de arranque, la etiqueta del sector de arranque de la partición activa es cargada y verificada.

22. Si la etiqueta de DOS del sector de arranque no puede ser leída exitosamente de la partición activa en cinco intentos debido a errores de lectura, el siguiente mensaje es desplegado y el sistema se detiene:

Error loading operating system

23. La etiqueta de DOS en el sector de arranque es revisada para encontrar una firma. Si la etiqueta de DOS en el sector de arranque no contiene el valor de firma de 55AAh como los dos últimos bytes en el sector, aparece el siguiente mensaje y el sistema se detiene:

Missing operating system

24. La etiqueta del sector de arranque es ejecutada como un programa, el cual revisa el directorio raíz para estar seguro de que los dos primeros archivos encontrados son IBMBIO.COM e IBMDOS.COM. Si estos archivos están presentes entonces son cargados.

25. Si el disco fue preparado con FORMAT o SYS usando DOS versión 3.3 o anterior, y en el sistema de especificación de archivos no se encuentran los primeros dos archivos de inicio en el directorio o si un problema fue encontrada al ser cargados, aparece el siguiente mensaje:

Non-System disk or disk error
Replace and strike any key when ready

26. Si el disco fue preparado con FORMAT o SYS usando DOS versión 3.3 o anterior y el sector de arranque está dañado, se obtiene el siguiente mensaje:

Disk Boot failure

27. Si el disco fue preparado con FORMAT o SYS usando DOS versión 4.0 o posterior, y en el sistema de especificación de archivos no encuentra los dos primeros archivos en el directorio, o si ocurrió un problema al cargarlos o el sector de arranque está dañado, aparece el siguiente mensaje:

Non-System disk or disk error
Replace and strike any key when ready

28. Si no ocurrieron problemas, la etiqueta de DOS del sector de arranque ejecuta el archivo IBMBIO.COM.

29. El código de inicialización de IBMBIO.COM se copia a si mismo dentro de la región alta de la memoria contigua de DOS y transfiere el control a la copia. La copia del código de inicialización relocaliza a IBMDOS sobre la porción de IBMBIO en memoria baja que contiene el código de inicialización, debido a que este código (que no es muy grande) necesita estar en esa localidad.

30. El código de inicialización ejecuta IBMDOS, la cual inicializa la base de los manejadores de dispositivos, determina el estado del equipo, realiza un reset en el sistema de discos, se da un reset e inicializa encendiendo los dispositivos y fija los parámetros por defecto del sistema.

31. El sistema de DOS se encuentra activo y el código de inicialización de IBMBIO obtiene nuevamente el control.

32. El código de inicialización de IBMBIO lee CONFIG.SYS cuatro veces.

33. Durante la primer lectura todas las declaraciones excepto DEVICE, INSTALL y SHELL son leídas y procesadas en orden predeterminado. De este modo, el orden de aparición de otras declaraciones diferentes a las anteriores mencionadas en CONFIG.SYS no son de importancia.

34. Durante la segunda lectura declaraciones tipo DEVICE son procesadas en el orden que aparezcan y si otros archivos de manejador de dispositivos son llamados, se cargan y se ejecutan.

35. Durante la tercer lectura las declaraciones INSTALL son procesadas en el orden que aparezcan, y los programas llamados son cargados y ejecutados.

36. Durante la cuarta y última lectura, la declaracion de SHELL es procesada y carga la especificación de procesador de comandos con la especificación de parámetros. Si el archivo CONFIG.SYS no contiene la declaración de SHELL, por defecto el procesador COMMAND.COM es cargado con parámetros por defecto. Cargando el procesador de comandos se sobreescribe el código de inicialización en memoria (porque el trabajo del código de inicialización se ha terminado).

37. Si AUTOEXEC.BAT está presente, COMMAND.COM es cargado y ejecuta AUTOEXEC.BAT. Después de haber sido ejecutados los comandos de AUTOEXEC.BAT el apuntador (prompt) de dos aparece (a menos que el AUTOEXEC.BAT llame a algún programa de aplicación o cubierta de algún tipo, en cuyo caso, el usuario puede operar el sistema sin haber visto el apuntador de DOS).

38. Si el AUTOEXEC.BAT no está presente, COMMAND.COM ejecuta los comandos internos de fecha y tiempo (DATE y TIME) desplegando un mensaje de derechos reservados (copyright) y despliega el apuntador de DOS.

Estos pasos son los primeros que ejecuta un sistema IBM AT y muchos sistemas IBM PC compatibles. Algunos cambios menores en estos pasos pueden ocurrir, tal como los introducidos por otros programas de ROM en varios adaptadores que pueden estar conectados en una ranura (slot); también depende de la exacta programación de ROM BIOS, puesto que algunos de los mensajes de error y secuencias pueden variar.

Además se puede modificar el procedimiento del sistema de encendido si se alteran los archivos CONFIG.SYS y AUTOEXEC.BAT. Estos archivos controlan la configuración de DOS y permiten una ejecución especial de programas para ser activados cada vez que el sistema da inicio.

APENDICE C: CODIGOS DE ERROR PARA IBM PC, PC/XT, PC PORTATILES Y COMPATIBLES

* Códigos de Error de POST

No hay beep y:

Pantalla en blanco	Problema con la fuente
No se puede leer la pantalla	Problema con la fuente
Mensaje de verificación de paridad	Problema con la memoria
Error 1XX	Problema con la tarjeta madre
1 beep largo y 1 beep corto	Reemplazar la tarjeta madre o los chips ROM BASIC
1 beep largo y 2 beeps cortos	Cada adaptador de video tiene falla o la tarjeta EGA tiene un probiema
1 beep largo y 3 beeps cortos	Cada adaptador de video tiene falla o la tarjeta EGA tiene un problema
1 beep corto y se depliega BASIC	Disco flexible o disco duro tienen problemas de arranque
1 beep corto y el prompt de DOS	Operación normal
1 beep corto y un beep largo	Problema con el adaptador de video

2 beeps cortos y:

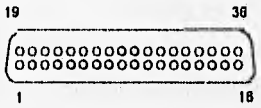
Pantalla en blanco o no se puede leer	Cada adaptador de video tiene falla o la tarjeta EGA tiene un problema
Imagen distorsionada	Cada adaptador de video tiene falla o la tarjeta EGA tiene un problema
Error Xabcd XX 201	Error de memoria en la tarjeta madre o en la tarjeta de expansión
Error XXXX 201	Error de memoria en la tarjeta madre o en la tarjeta de expansión
Error 301	Problema con el teclado
Error XX 301	Problema con el teclado
	Problema con el código de rastreo de una tecla específica
Error 601	Problema con el drive
Error 17XX	Problema con el disco duro o con el adaptador
Error 30XX	Problema con el adaptador primario de PC para red
Error 31XX	Problema con el adaptador alterno de PC para red

Error C80000 ROM	Reemplazar el adaptador del disco duro
Error CC000 ROM	Problema con el adaptador primario de PC para red
Error de ROM	Reemplazar la tarjeta madre
Error FXXXX ROM	Reemplazar la tarjeta madre
Beeps continuos	Problema con la fuente
Repetición de beeps cortos	Problema con la fuente

APENDICE D: INTERFACES Y DIAGRAMAS DE CIRCUITOS

*** Interfaz Paralelo Centronics**

Conectores de interfase:



Impresora: Amphenol 57LE-40360 o equivalente
Cable: Amphenol 57FE-30360 o equivalente
Tipo cable: Cable torcido de pareja
Longitud: 1.9m (6 ft) máximo

Pin	Señal	Pin	Señal
1	Data Strobe	19	Tierra
2	Bit data 0	20	
3	Bit data 1	21	
4	Bit data 2	22	
5	Bit data 3	23	
6	Bit data 4	24	
7	Bit data 5	25	
8	Bit data 6	26	
9	Bit data 7	27	
10	ACKNLG	28	
11	Busy	29	
12	PE	30	Input Prime Ret
13	+5 V (4.7K Ohms sacar)	31	Input Prime
14	Outo Feed	32	Fault
15	No usado	33	0 V
16	0 V	34	No usado
17	0V	35	+5 V (4.7K Ohms sacar)
18	+5 V	36	SLCT IN

*** Descripción de señales**

DATA STROBE: (Entrada)

Pulso de estroboscopio de lectura dato cuya amplitud minima debe de ser 1ms. Alto (high) en estado estable, el dato es leído al flanco anterior cuando este signo es bajo (low).

DATA 0-7: (Entrada)

Desde el bit 0 hasta bit 7 del dato. El nivel es High (alto) para 1 y Low (bajo) para 0. La amplitud mínima debe ser de 3ms.

ACKNLG: (Salida)

Pulso de reconocimiento de salida al completamiento de la entrada de dato o al completamiento de la operación de la impresora. La amplitud del pulso mínima, 6ms.

BUSY: (Salida)

Señal para indicar si la impresora puede aceptar datos. La entrada de dato es posible cuando esta señal se coloca en el nivel bajo (low).

Esta señal va al nivel alto (high) bajo las siguientes condiciones:

a) Durante la operación de la impresora
b) Durante la entrada del dato
c) En el estado de fuera de linea (Deselect)

PE (Paper End): (Salida)

Señal del nivel de DC que va al nivel alto (high) cuando el estado vacío de papel es descubierto.

+5V (Salida)

Elevar hasta +5V por resistor de 4.7K Ohms

AUTO FEED: (Entrada)

Cuando esta señal va al nivel bajo (Low), la impresora carga automáticamente el papel.

+5V SOURCE: (Salida)

El máximo de la salida es 300mA

INPUT PRIME: (Entrada)

Cuando esta señal va al nivel bajo (low), la impresora es reiniciada con los valores iniciales.

FAULT: (Salida)

Señal del nivel de DC que va al nivel bajo (low) cuando la impresora está en el estado de fuera de línea.

SLCT IN: (Entrada)

Cuando esta señal va al nivel bajo (low) coloca a la impresora en el estado de en línea, en caso contrario, fuera de línea.

- Circuito de interfaz

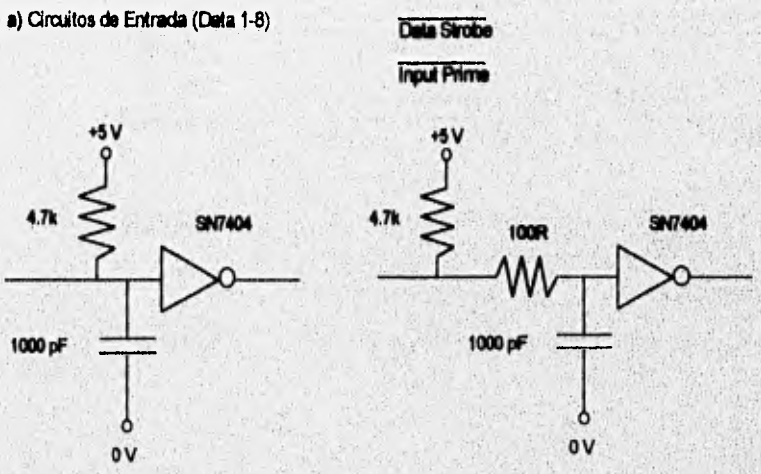
Las señales de la línea de interface son compatibles a los niveles lógico de TTL. Los niveles de entrada y salida son como se muestran a continuación:

	Nivel Bajo (Low) V	Nivel alto (High) V
Entrada	0 <= V <= 0.4	2.4 <= V <= 5.0
Salida	0 <= V <= 0.4	2.4 <= V <= 5.0

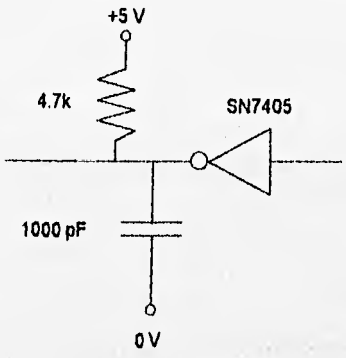
En las siguientes figuras se muestran los diagramas de interfases compatibles al nivel lógico TTL:

- Circuitos de Entrada:

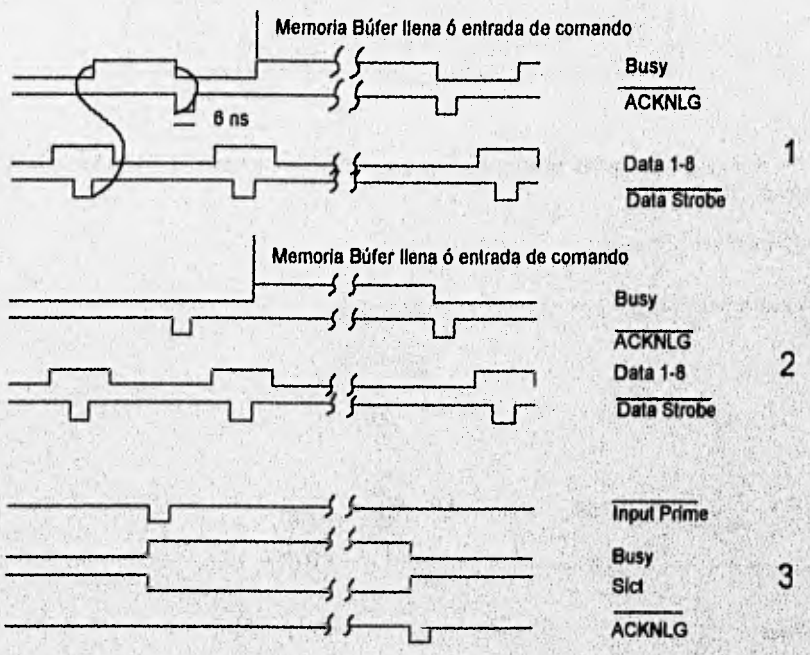
a) Circuitos de Entrada (Data 1-8)



- Circuito de salida:



Tablas de distribución:



1) Dato aceptado bajo la condición de un dato BUSY
2) Data aceptado bajo la condición de una línea BUSY
3) Dato aceptado de INPUT PRIME

*** Interfaz serial (RS-232)**

Este tipo de interfaz es un estándar para conectar componentes de sistemas tales como: modems, impresoras y computadoras. El estándar fue establecido por Electronic Industries Association (EIA).

El RS-232 cuenta con 25 hilos de señal (pines) que establecen 18 circuitos con un retorno a través de tierra. Además, el estándar define los voltajes utilizados en todos los circuitos (los rangos para 0 y 1 lógicos).

El estándar fue definido por el CCITT (Comité Consultivo Internacional de Telefonía y Telegrafía), dando como resultado una interfaz que satisface completamente a la industria de este ramo. El puerto serial de la IBM PC solamente utiliza nueve hilos y puede funcionar solo con tres de ellos.

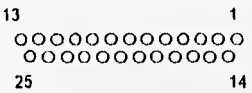
Un voltaje entre 2 y 5 volts es un 1 lógico, y un voltaje entre 0 y 0.8 volts es un 0 lógico. Estos niveles no son utilizados fuera de la computadora, debido a que ellos no proveen de la suficiente inmunidad contra ruido eléctrico.

La computadora y muchos equipos de comunicaciones tienen señales de salida de ±12 volts. Sin embargo, una entrada de más o menos 3 volts es suficiente para definir el estado lógico.

La señal de la RS-232 es reversible en polaridad proveniente de las señales TTL que maneja la computadora; por ejemplo, 3 volts TTL es un 1 lógico que es equivalante a -12 volts, 1 lógico o marca.

- **Conectores de interfaz:**

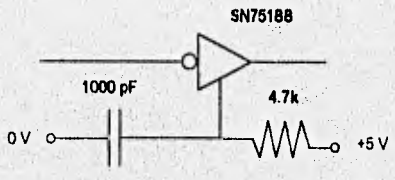
Impresora:	Amphenol 17LE-13250 o equivalente
Cable:	Amphenol 17JE-23250 o equivalente
Tipo cable:	Blindado
Longitud:	2 m (6.5 ft) máximo



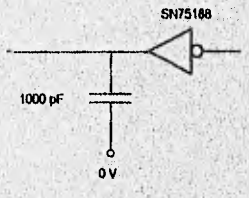
PIN	Señal	IN/OUT	PIN	Señal	IN/OUT
1	FG		14	SCA	Ø
2	SD	Ø	15	NC	
3	RD	×	16	NC	
4	RTS	Ø	17*	Vientre de intensidad de corriente (DTR)	Ø
5	CTS	×	18	NC	
6	DSR	×	19	NC	
7	SG		20	DTR	Ø
8	CD	×	21	NC	
9	NC		22	NC	
10	NC		23*	Vientre de intensidad de corriente (RP) RET	
11	SCA	Ø	24*	Vientre de intensidad de corriente (DTR) RET	
12	NC		25*	Vientre de intensidad de corriente (RP)	×
13	NC				

Nota: Ø = Out, x = In, son las direcciones de entrada/salida de las señales
* Las señales de los pines 17, 23, 24 y 25 son opcionalmente disponibles

- **Circuitos de interfaz:**



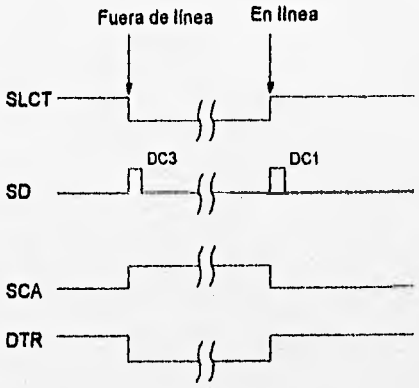
a) Circuito de entrada



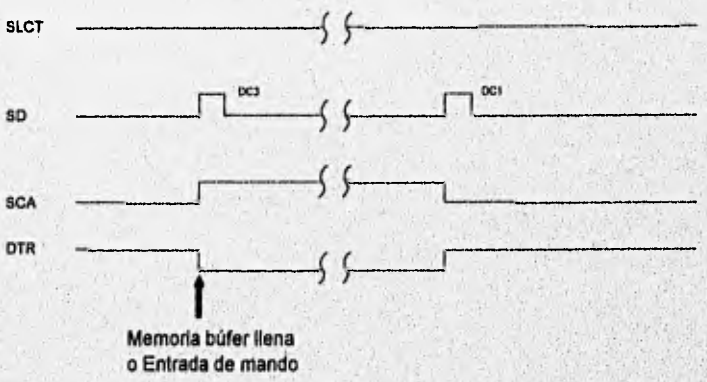
b) Circuito de salida

Tablas de distribución

1) Giro a modos de fuera de línea y de en línea

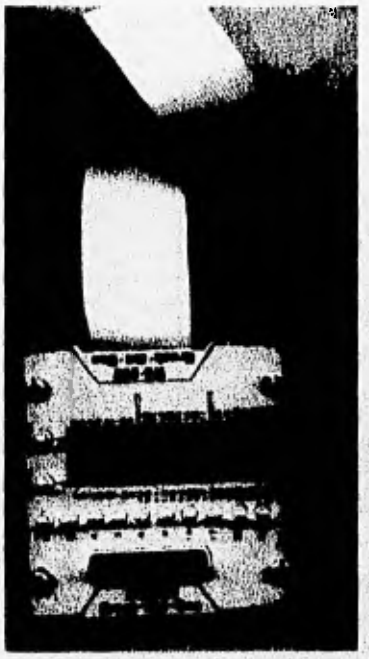


2) Dato aceptado



APENDICE E: CONECTOR RS-232 LOOPBACK

La figura que a continuación se muestra es un conector serial para realizar la prueba de LoopBack (Retroalimentación) utilizando un conector DB25. Este tipo de conector es versátil en la forma en que puede ser configurado, debido a que solo utilizando puentes (alambres) se pueden realizar las conexiones adecuadas para el LoopBack.



Conector RS-232 Break-Out Bin

Si se desea construir algún tipo de conector para realizar las pruebas de LoopBack, se deben de tomar en cuenta las características que se describen en las tablas siguientes:

- ***Conector de Puerto Serial 9 pines (AT)***

Pin	Descripción	Señal	Dirección
1	Carrier Detect	CD	In
2	Receive Data	RD	In

Pin	Descripción	Señal	Dirección
3	Transmit Data	TD	Out
4	Data Terminal Ready	DTR	Out
5	Signal Ground	SG	-
6	Data Set Ready	DSR	In
7	Request to Send	RTS	Out
8	Clear to Send	CTS	In
9	Ring Indicator	RI	In

- Conector de Puerto Serial 25 pines (PC, XT y PS/2)

Pin	Descripción	Señal	Dirección
1	Chassis Ground	-	-
2	Transmit Data	TD	Out
3	Receive Data	RD	In
4	Request to Send	RTS	Out
5	Clear to Send	CTS	In
6	Data Set Ready	DSR	In
7	Signal Ground	SG	-
8	Carrier Detect	CD	In
9	(+) Transmit current Loop Return	-	Out
11	(-) Transmit current Loop Data	-	Out
18	(+) Receive current Loop Data	-	In
20	Data Terminal Ready	DTR	Out
22	Ring Indicator	RI	In
25	(-) Receive current Loop Return	-	In

Nota: Pines 9, 11, 18 y 25 son usados para una retroalimentación en la interfaz.

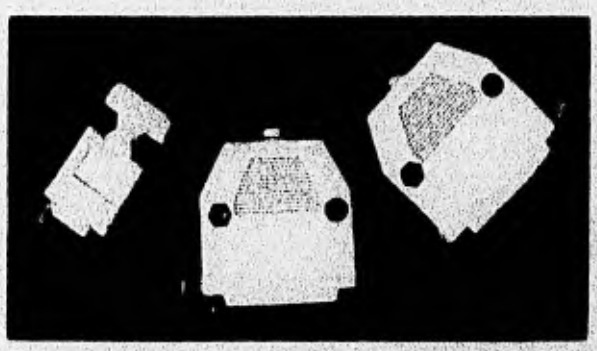
- Conector de Puerto Paralelo 25 pines (PC,XT, AT y PS/2)

Pin	Descripción	Dirección
1	- Strobe	Out
2	+ Data Bit 0	Out
3	+Data Bit 1	Out
4	+Data Bit 2	Out
5	+ Data Bit 3	Out
6	+ Data Bit 4	Out
7	+Data Bit 5	Out
8	+ Data Bit 6	Out
9	+ Data Bit 7	Out

Pin	Descripción	Dirección
10	- Acknowledge	In
11	+ Busy	In
12	+Paper End	In
13	+ Select	In
14	- Auto Feed	Out
15	- Error	In
16	- Initialize Printer	Out
17	- Select Input	Out
18	- Data Bit 0 Return (Ground)	In
19	- Data Bit 1 Return (Ground)	In
20	- Data Bit 2 Return (Ground)	In
21	- Data Bit 3 Return (Ground)	In
22	- Data Bit 4 Return (Ground)	In
23	- Data Bit 5 Return (Ground)	In
24	- Data Bit 6 Return (Ground)	In
25	- Data Bit 7 Return (Ground)	In

- Conexiones para cable serial adaptador de 9 pines a 25 pines

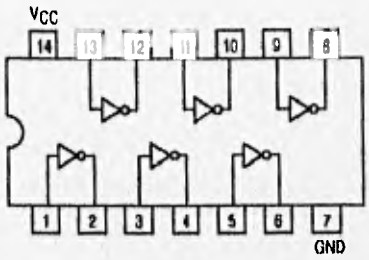
Descripción	Señal	9 pines	25 pines
Carrier Detect	CD	1	8
Receive Data	RD	2	3
Transmit Data	TD	3	2
Data Terminal Ready	DTR	4	20
Signal Ground	SG	5	7
Data Set Ready	DSR	6	6
Request to Send	RTS	7	4
Clear to Send	CTS	8	5
Ring Indicator	RI	9	22



Conectores LoopBack RS-232/DB17 (IBM 72X8546)

APENDICE F: ESPECIFICACION DE CIRCUITOS

74LS04 HEX INVERTER



Guaranteed Operating Ranges

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current - High			-0.4	mA
I_{OL}	Output Current - Low			8.0	mA

DC Characteristics over operating temperature range

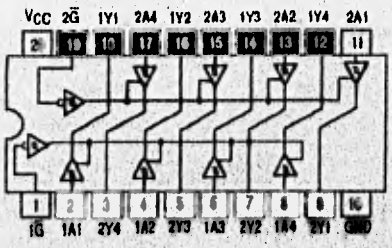
Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{IH}	Input High Voltage	2.0			V
V_{IL}	Input Low Voltage			0.8	V
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V
V_{OH}	Output High Voltage	2.7	3.5	0.5	V
V_{OL}	Output Low Voltage		0.25	0.4	V
			0.35	0.5	V

I_{IH}	Input High Current			20 0.1	mA mA
I_{IL}	Input Low Current			-0.4	mA
I_{OS}	Short Circuit Current	-20		-100	mA
I_{CC}	Power Supply Current Total, Output High			2.4	mA
	Total Output Low			6.6	

74LS244 OCTAL BUFFER/LINE DRIVER WITH 3-STATE OUTPUTS

Tabla de Verdad

INPUTS		OUPUT
$\overline{1G}$, $\overline{2G}$	D	
L	L	L
L	H	H
H	X	(Z)



H=High Voltage Level
L=Low Voltage Level
X=Immaterial
Z=High Impedance

Guaranteed Operating Ranges

Symbol	Parameter	Min	Typ	Max	Unit
Vcc	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current - High			-15	mA
I_{OL}	Output Current - Low			24	mA

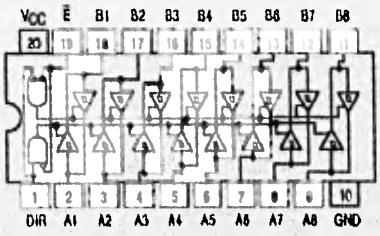
DC Characteristics over operating temperature range

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{IH}	Input High Voltage	2.0			V
V_{IL}	Input Low Voltage			0.8	V
V_{T+}-V_{T-}	Hysteresis	0.2	0.4		
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V
V_{OH}	Output High Voltage	2.0			V
V_{OL}	Output Low Voltage		0.25	0.4	V
			0.35	0.5	V
I_{OZH}	Output Off Current High			20	mA
I_{OZL}	Output Off Current Low			-20	mA
I_{IH}	Input High Current			20	mA
				0.1	mA
I_{IL}	Input Low Current			-0.2	mA
I_{OS}	Short Circuit Current	-40		-225	mA
I_{CC}	Power Supply Current Total, Output High			27	mA
	Total, Output Low			46	
	Total at High Z			54	

74LS245 OCTAL BUS TRANSCEIVER

Tabla de Verdad

INPUTS		OUPUT
$\overline{E}$	D	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	Isolation



Guaranteed Operating Ranges

Symbol	Parameter	Min	Typ	Max	Unit
Vcc	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current - High			-15	mA
I_{OL}	Output Current - Low			24	mA

DC Characteristics over operating temperature range

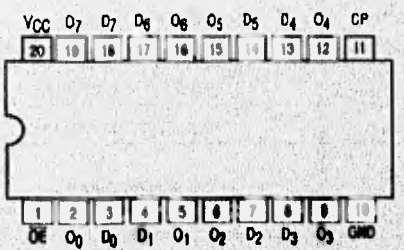
Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{IH}	Input High Voltage	2.0			V
V_{IL}	Input Low Voltage			0.8	V
V_{T+}-V_{T-}	Hysteresis	0.2	0.4		
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V
V_{OH}	Output High Voltage	2.0			V
V_{OL}	Output Low Voltage		0.25	0.4	V
			0.35	0.5	V
I_{OZH}	Output Off Current High			20	mA

I_{OZL}	Output Off Current Low				-200	mA
I_{IH}	Input High Current	A or B, $\overline{DR}$ or E			20	mA
		$\overline{DR}$ or E			0.1	mA
		A or B			0.1	mA
I_{IL}	Input Low Current				-0.2	mA
I_{OS}	Short Circuit Current		-40		-225	mA
I_{CC}	Power Supply Current Total, Output High				70	mA
	Total, Output Low				90	
	Total at High Z				95	

74LS374 OCTAL D TYPE FLIP-FLOP WITH 3-STATE OUTPUT

Tabla de Verdad

D_n	LE	$\overline{OE}$	O_n
H	⎍	L	H
L	⎍	L	L
X	X	H	Z*



H=High Voltage Level
L=Low Voltage Level
X=Immaterial
Z=High Impedance

*Note: Contents of flip-flops unaffected by the state of the Output Enable Input ($\overline{OE}$).

Guaranteed Operating Ranges

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current - High			-2.6	mA
I_{OL}	Output Current - Low			24	mA

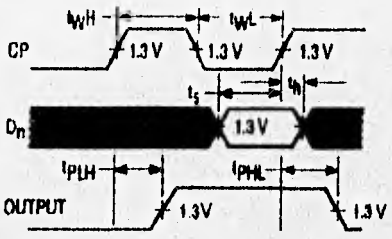
DC Characteristics over operating temperature range

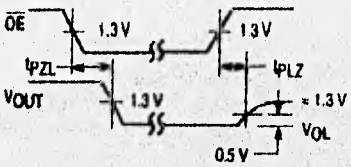
Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{IH}	Input High Voltage	2.0			V
V_{IL}	Input Low Voltage			0.8	V
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V
V_{OH}	Output High Voltage	2.4	3.1		V
V_{OL}	Output Low Voltage		0.25	0.4	V
			0.35	0.5	V
I_{OZH}	Output Off Current High			20	mA
I_{OZL}	Output Off Current Low			-20	mA
I_{IH}	Input High Current			20	mA
				0.1	mA
I_{IL}	Input Low Current			-0.4	mA
I_{OS}	Short Circuit Current	-30		-130	mA
I_{CC}	Power Supply Current			40	mA

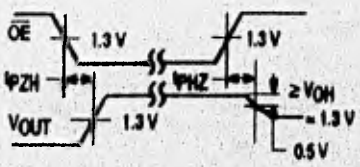
AC Characteristics (T_A=25 °C, Vcc=5.0 V)

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
f_{MAX}	Maximum Clock Frequency	35	50		MHz
t_{PLH} t_{PHL}	Propagation Delay, Data to Output				ns
t_{PLH} t_{PHL}	Clock or Enable to Output		15 19	28 28	ns
t_{PZH} t_{PZL}	Output Enable Time		20 21	28 28	ns
t_{PHZ} t_{PLZ}	Output Disable Time		12 15	20 25	ns

Formas de Onda de AC

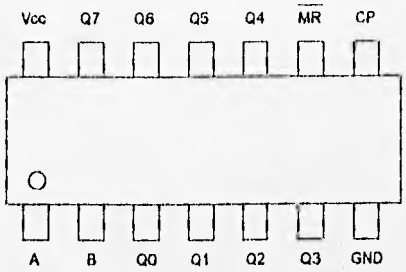






74LS164 SERIAL IN PARALLEL OUT SHIFT REGISTER

Modo de Selección y Tabla de Verdad



Operating Mode	INPUTS			OUTPUTS	
	$\overline{MR}$	A	B	Q_0	Q_1-Q_7
Reset (Clear)	L	X	X	L	L-L
Shift	H	l	l	L	q_0-q_6
	H	l	h	L	q_0-q_6
	H	h	l	L	q_0-q_6
	H	h	h	H	q_0-q_6

Pin Names

A,B	Data Inputs
CP	Clock (Active High going edge) Input
$\overline{MR}$	Master Reset (Active Low) Input
Q_0-Q_7	Outputs

Guaranteed Operating Ranges

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply Voltage	4.75	5.0	5.25	V
T_A	Operating Ambient Temperature Range	0	25	70	°C
I_{OH}	Output Current - High			-0.4	mA
I_{OL}	Output Current - Low			8.0	mA

DC Characteristics over operating temperature range

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{IH}	Input High Voltage	2.0			V
V_{IL}	Input Low Voltage			0.8	V
V_{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V
V_{OH}	Output High Voltage	2.7	3.5		V
V_{OL}	Output Low Voltage		0.25	0.4	V
			0.35	0.5	V
I_{IH}	Input High Current			20	mA
				0.1	mA
I_{IL}	Input Low Current			-0.4	mA
I_{OS}	Short Circuit Current	-20		-100	mA
I_{CC}	Power Supply Current			27	mA

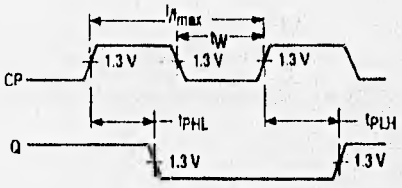
AC Characteristics (T_A=25 °C)

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
f_{MAX}	Maximum Clock Frequency	25	36		MHz
t_{PLH} t_{PHL}	Propagation Delay, $\overline{MR}$ to output Q		24	36	ns
t_{PLH} t_{PHL}	Propagation Delay Clock to Output Q		17 21	27 32	ns

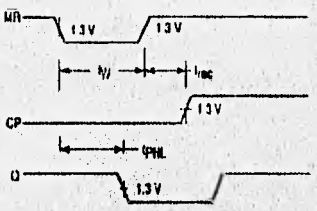
AC SETUP REQUIRMENTS (T_A=25°C)

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
t_W	CP, MR Pulse Width	20			ns
t_s	Data Setup Time	15			ns
t_h	Data Hold Time	5.0			
t_{rec}	MR to Clock Recovery Time	20			ns

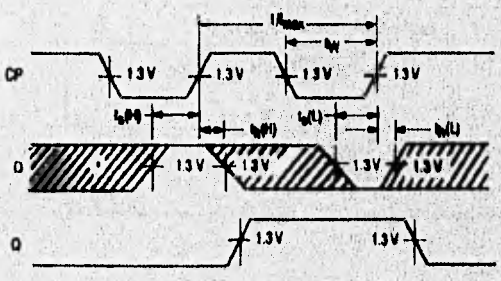
Formas de Onda de AC



Clock to Output Delays and Clock Pulse Width

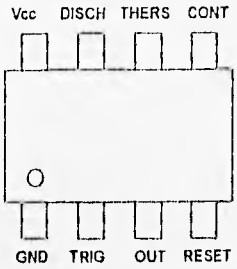


Master Reset Pulse Width, Master Reset to Output Delay and Master Reset to Clock Recovery Time



Data Setup and Hold Times

SE555 Precision Timer



Recommended Operating Conditions

	SE555		Unit
	Min	Max	
Supply voltage, Vcc	4.5	18	V
Input voltage (control, reset, threshold and trigger)	Vcc	Vcc	V
Output current	±200	±200	mA
Operating free-air Temperature, T_A	-55	125	°C

Electrical characteristics at 25°C free air temperature, Vcc=5 V to 15 V

Parameter	Test Conditions	Min Typ Max	Unit
Threshold voltage level	Vcc= 15 V	9.4 10 10.6	V
	Vcc= 5 V	2.7 3.3 4	
Threshold current		30 250	nA
Trigger voltage level	Vcc= 15 V	4.8 5 5.2	V
	Vcc= 5 V	1.45 1.7 1.9	

Trigger current	Trigger at 0 V		0.5 0.9	mA
Reset voltage level			0.3 0.7 1	V
Reset current	Reset at Vcc		0.1 0.4	mA
	Reset at 0 V		-0.4 -1	
Discharge switch off-state current			20 100	nA
Control voltage (open circuit)	Vcc= 15 V		9.6 10 10.4	V
	Vcc= 5 V		2.9 3.3 3.8	
Low-level output voltage	Vcc= 15 V	I_{OL}= 10 mA	0.1 0.15	V
		I_{OL}= 50 mA	0.4 0.5	
		I_{OL}= 100 mA	2 2.2	
		I_{OL}= 200 mA	2.5	
	Vcc= 5 V	I_{OL}= 5 mA	0.1 0.2	
		I_{OL}= 8 mA	0.15 0.25	
High-level output voltage	Vcc= 15 V	I_{OH}= -100 mA	13 13.3	V
		I_{OH}= -200 mA	12.5	
	Vcc= 5 V	I_{OH}= -100 mA	3 3.3	
Supply current	Output Low, No Load	Vcc= 15 V	10 12	mA
		Vcc= 5 V	3 5	
	Output High, No Load	Vcc= 15 V	9 10	
		Vcc= 5 V	2 4	

INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento

CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

CONCLUSIONES

Antes de iniciar el desarrollo del sistema, se buscaron antecedentes del funcionamiento de las computadoras e impresoras con el fin de entender más a fondo la operación de estos dispositivos y así poder identificar las fallas que se presentan en el hardware y el software.

Al conocer como operan los equipos de cómputo (procesamiento, manejo de interrupciones, almacenamiento, etc.) se ideo la forma de presentar adecuadamente el sistema de diagnóstico, así el programa diseñado proporciona la información del equipo que se este analizando (configuración básica, verificación de impresión, etc.).

Para el desarrollo del programa, la información presentada se obtuvo con ayuda del programa DEBUG (MS-DOS). Con este programa se consultaron las direcciones y bytes en ROM BIOS que proporcionan la configuración del equipo, la cual fue confirmada con los datos incluidos en los manuales o lista de configuración proporcionados por el fabricante de cada computadora en que se trabajo y con la utilización de software comercial (Norton Utilities y CheckIt).

La forma en que se presentan los pasos para la obtención de la configuración del equipo que se este analizando (capítulo V), es la que nosotros utilizamos para poder interpretar la información obtenida; además, pensamos que con este tipo de presentación el lector pueda comprobar y llevar a cabo las pruebas que nosotros realizamos, pero de una manera más sencilla.

Gran parte de la información contenida en el ROM BIOS no se encuentra documentada por lo que muchos de estos datos no pueden ser interpretados correctamente; por ejemplo, existen datos tales como: identificador y velocidad del microprocesador, características de disco duro, etc.

Uno de los problemas que nos encontramos al tratar de obtener la configuración completa de una computadora, fue como manejar y obtener información acerca de las memorias expandida y extendida, este tipo de memorias son manejadas mediante controladores proporcionados por diseñadores de software, los cuales no publican o restringen el acceso a la información sobre su manejo. Si se conocieran tales datos, se podría implementar y mejorar un método de diagnóstico de las páginas de almacenamiento y la existencia de alguna falla en estos tipos de manejadores de memoria.

La presentación de la configuración del equipo proporcionada por el programa de diagnóstico, es para que el usuario al detectar algún tipo de falla en su equipo, verifique los datos proporcionados por el fabricante y los reales obtenidos mediante el software, y de esta manera localizará adecuadamente el origen de la falla; esto se facilita a través de la consulta de una lista de fallas y posibles causas, proporcionadas en el texto.

En lo que se refiere al manejo y operación de impresoras es difícil conformar un estándar, debido a que los protocolos existentes entre las diferentes clases de estas (matriz de puntos, inyección de tinta, láser, etc) no son compatibles entre ellos; por lo que en este caso, se tomó la decisión de manejar una sola clase de impresora (matriz de puntos de 9 agujas en modo

EPSON), con la que podemos dar a conocer las bases del funcionamiento de las impresoras en una forma general, así como el manejo de los protocolos.

En el manejo de impresión encontramos dos formas de escribir al puerto: por medio de DOS o escribiendo en la dirección base del puerto (BIOS); resultando esta última más conveniente para el diagnóstico del puerto utilizado. Por otro lado, debemos mencionar que debido a la diferencia de protocolos y forma de operación de las impresoras, el byte de estado almacenado en la dirección base es interpretado de diferentes formas para el diseño de controladores de impresión.

Las pruebas de análisis desarrolladas con el programa, conectores y simulador, además de la información presentada, nos ayuda a determinar fallas comunes que puede presentar un equipo de cómputo; este tipo de análisis puede llegar a ser más complicado conforme las computadoras evolucionen y no se tenga información concerniente a las características de estas, contenidas en ROM BIOS o en algún otro dispositivo

El diseño e implementación de este sistema cubre algunos de los requerimientos de programas comerciales, ya mencionados; presentando desde luego la información de una forma más accesible para las personas que no se encuentren familiarizadas con sistemas de cómputo.

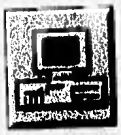
INTRODUCCION

CAPITULO I: Sistemas Operativos

CAPITULO II: Software y Hardware

CAPITULO III: Impresoras y Protocolos

CAPITULO IV: Análisis de Rendimiento



CAPITULO V: Diseño e Implementación de Interfaces

APENDICES

CONCLUSIONES

BIBLIOGRAFIA

BIBLIOGRAFIA

Upgrading and Reparing PC's
2nd Edition
Scott Mueller
Editorial QUE Corporation
1992
USA

Build Your Own 80486 PC and Save a Bundle
1rst Edition
Aubrey Pilgrin
Windcrest/McGrawHill
1991
USA

Solucionario del Programador para IBM PC, XT, AT y compatibles
Primera edición
Robert Jourdain
Editorial Anaya Multimedia
1990
España

Computer Interfacing, connection to the real world
1rst Edition
Martin Cripps
Editorial Edward Arnold
1989
Great Britain

Guide to IBM PC Communications
1rst Edition
David Kruglinski
Editorial The Osborne/McGrawHill
1984
USA

The New Peter Norton Programmer's Guide to the IBM PC & PS/2
1rst Edition
Peter Norton, Richard Wilton
Microsoft Press
1988
USA

PCMagazine DOS Power Tools, Techniques, Tricks and Utilities
2nd Edition
Editorial PCMagazine
1990
USA

The Intel Microprocessors 8086/8088, 80186, 80286, 80386, and 80486 Architecture, Programming and Interfacing
2nd Edition
Barry B. Brey
Editorial Macmillan Publishing Company
1991
USA

Impresoras Matriciales, chorro de tinta y láser
Juan Carlos Oros
1a Edición
Editorial Paraninfo
1991
España

Compufacts Tips, Tricks, Facts, and Secrects for Resolving Software and Hardware Problems
Ronald J. Dugreiner
1rst Edition
Editorial Compufacts
1991
USA

Impresora de Matriz de Impacto M-1509, Manual del Usuario
Brother
1a Edición
Brother
1986
Japón

Angel Bosch Torrano
"Controladores CRT"
Electrónica Hoy, Revista Profesional de Electrónica y Comunicaciones
México D.F.
No 4, Agosto 1992
p.p. 55-58

Angel Bosch Torrano
"Curso de Electrónica Digital"
Suplemento No 14
Electrónica Hoy, Revista Profesional de Electrónica y Comunicaciones
México D.F.
No 6, Junio 1993
p.p. 189-190

Angel Bosch Torrano
"Curso de Electrónica Digital"
Suplemento No 15
Electrónica Hoy, Revista Profesional de Electrónica y Comunicaciones
México D.F.
No 7, Julio 1993
p.p. 191-196

Angel Bosch Torrano
"Curso de Electrónica Digital"
Suplemento No 16
Electrónica Hoy, Revista Profesional de Electrónica y Comunicaciones
México D.F.
No 8, Agosto 1993
p.p. 197-198

Angel Bosch Torrano
"Computadoras, Manual de Aprendizaje Rápido No 4"
Electrónica Práctica-Resistor
México D.F.
No 12, Diciembre 1993
p.p. 19-25

Angel Bosch Torrano
"Sabía Ud. Que: MEMORIAS FLASH, un nuevo concepto en almacenamiento digital"
Primera parte
Electrónica Práctica-Resistor
México D.F.
No 12, Diciembre 1993
p.p. 65-69

Angel Bosch Torrano
"Computadoras, Manual de Aprendizaje Rápido No 5"
Electrónica Práctica-Resistor
México D.F.
No 1, Enero 1994
p.p. 34-37

Angel Bosch Torrano
"Sabía Ud. Que: MEMORIAS FLASH, un nuevo concepto en almacenamiento digital"
Segunda parte
Electrónica Práctica-Resistor
México D.F.
No 1, Enero 1994
p.p. 49-51

Angel Bosch Torrano
"Sabía Ud. Que: Los TFT LCD's para uso específico en computadoras notebooks"
Primera parte
Electrónica Práctica-Resistor
México D.F.
No 6, Junio 1994
p.p. 47-48

Angel Bosch Torrano
"Sabía Ud. Que: Los TFT LCD's para uso específico en computadoras notebooks"
Segunda parte
Electrónica Práctica-Resistor
México D.F.
No 7, Julio 1994
p.p. 45-48

Santiago J. Villazón
"Monitores: 1024 de Color; Mayor Resolución: Todo está en el tiempo"
PC Magazine en Español
México D.F.
No 4, Julio 1990
pág. 24

Angel Bosch Torrano
"Monitores de 14 y 15 pulgadas: Opciones para todos"
PC Magazine en Español
México D.F.
No 6, Junio 1993
p.p. 67-68

Angel Bosch Torrano
"Monitores de 14 y 15 pulgadas: ¿Cómo trabaja un monitor?"
PC Magazine en Español
México D.F.
No 6, Junio 1993
pág. 81

Angel Bosch Torrano
"Anatomía Lógica de un Diskette"
PC Magazine en Español
México D.F.
No 6, Junio 1993

Angel Bosch Torrano
"Tecnología de PC's: Dentro del Pentium"
PC Magazine en Español
México D.F.
No 7, Julio 1993
p.p. 83-90

Santiago J. Villazón
"Plataformas: Las Nuevas PC's; Comparando Arquitecturas de CPU"
PC Magazine en Español
México D.F.
No 7, Julio 1993
p.p. 62-63

Santiago J. Villazón
"Pista: CD-ROM"
PC Magazine en Español
México D.F.
No 8, Agosto 1993

Santiago J. Villazón
"CD-ROM, casi un disco de arena"
PC Magazine en Español
México D.F.
No 2, Febrero 1994
p.p. 83-85

Santiago J. Villazón
"4to número anual de impresoras"
PC Magazine en Español
México D.F.
No 3, Marzo 1994
p.p. 23-82