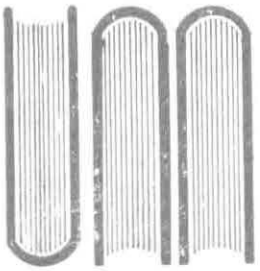
UNIVERSIDAD DE MONTERREY

DIVISION DE INGENIERIA Y CIENCIAS NATURALES Y EXACTAS



SISTEMA PARA LA CONVERSION
DE FRASES ASCII A VOZ ELECTRONICA

REPORTE DEL PROGRAMA DE EVALUACION FINAL

QUE PRESENTA

GONZALO EDUARDO FREILE CARDOZO

EN OPCION AL TITULO DE

INGENIERO EN COMPUTACION ADMINISTRATIVA
Y DE PRODUCCION

SAN PEDRO, GARZA GARCIA, N. L.
DICIEMBRE DE 1991

A mis padres
por su paciencia y cariño infinitos

Madre
Gracias por depositar tu fé en mi

Padre
Gracias por tu apoyo incondicional

SISTEMA PARA LA CONVERSION DE FRASES ASCII A VOZ ELECTRONICA

INDICE

CAPITULO I

INTRODUCCION..... 1

I.1. OBJETIVO DEL PROYECTO..... 1

I.2. ANTECEDENTES(DATOS PRELIMINARES)..... 2

I.3. JUSTIFICACION..... 3

I.4. DESARROLLO DEL PROYECTO..... 4

I.5. CONVENCIONES UTILIZADAS EN ESTE REPORTE..... 6

I.6. PANORAMA GENERAL DEL REPORTE ESCRITO..... 7

CAPITULO II

CONCEPTOS BASICOS DEL SISTEMA OPERATIVO (DOS)..... 9

II.1. HISTORIA DEL DOS..... 10

II.2. COMPONENTES DEL SISTEMA DOS..... 12
II.2.1. EL CARGADOR DEL SISTEMA OPERATIVO..... 12
II.2.2. EL BIOS..... 13
II.2.3. EL NUCLEO DEL DOS..... 14
II.2.3.1. Control de procesos..... 14
II.2.3.2. Manejo de memoria..... 15
II.2.3.3. Soporte periférico..... 15
II.2.3.4. El sistema de archivos..... 15
II.2.4. LA INTERFASE DEL USUARIO (SHELL)..... 16
II.2.5. PROGRAMAS DE SOPORTE..... 17

II.3. ESTRUCTURA DEL DOS..... 17

II.4. INTERFASE DEL PROGRAMADOR HACIA EL DOS..... 20

II.5. ¿ COMO SE CARGA EL DOS ?..... 21

CAPITULO III

MANEJADORES DE INTERRUPCIONES (PROGRAMAS TSR)..... 25

III.1. ¿ QUE ES UNA INTERRUPCION ?..... 26

III.2. TIPOS DE INTERRUPCIONES..... 27
III.2.1. INTERRUPCIONES INTERNAS DEL HARDWARE..... 27
III.2.2. INTERRUPCIONES EXTERNAS DEL HARDWARE..... 27
III.2.3. INTERRUPCIONES DE SOFTWARE..... 29

III.3. LA TABLA DEL VECTOR DE INTERRUPCIONES..... 31

III.4. COMO TRABAJAN LAS INTERRUPCIONES..... 31

III.5. ¿ QUE ES UN PROGRAMA TSR ?..... 34

III.6. ESTRUCTURA DE UN PROGRAMA TSR..... 34

III.7. TIPOS DE PROGRAMAS TSR..... 36
III.7.1. PROGRAMAS TSR ACTIVOS..... 37
III.7.2. PROGRAMAS TSR PASIVOS..... 37
III.7.3. PROGRAMAS TSR HIBRIDOS..... 37

III.8. ¿ CUANDO UTILIZAR UN PROGRAMA TSR ?..... 38

III.9. LOS MECANISMOS PARA LA CONSTRUCCION
DE UN PROGRAMA TSR..... 39

CAPITULO IV

CONDUCTORES DE DISPOSITIVOS (DEVICE DRIVERS)..... 42

IV.1. DISPOSITIVOS DE LAS COMPUTADORAS PERSONALES..... 45

IV.2. JERARQUIA DE LOS DEVICE DRIVERS..... 47

IV.3. POSIBLES PRESENTACIONES DE LOS DEVICE DRIVERS..... 49
IV.3.1. DEVICE DRIVERS PARA NUEVOS DISPOSITIVOS..... 49
IV.3.2. DEVICE DRIVERS DE REEMPLAZO PARA
DISPOSITIVOS ESTANDAR..... 50
IV.3.3. DEVICE DRIVERS SIN DISPOSITIVO..... 50

IV.4. TIPOS DE DEVICE DRIVERS..... 51
IV.4.1. DEVICE DRIVERS DE CARACTERES..... 51
IV.4.2. DEVICE DRIVERS DE BLOQUES..... 54

IV.5. ESTRUCTURA DE LOS DEVICE DRIVERS..... 55
IV.5.1. DEVICE HEADER..... 56
IV.5.2. ALMACENAMIENTO DE DATOS Y PROCEDIMIENTOS LOCALES.... 58
IV.5.3. PROCEDIMIENTO DE ESTRATEGIA..... 59

IV.5.4. PROCEDIMIENTO DE INTERRUPCION..... 60
IV.5.5. PROCESAMIENTO DE COMANDOS..... 62
IV.5.5.1. Inicialización..... 63
IV.5.5.2. Verificación del medio..... 64
IV.5.5.3. Construir BPB..... 64
IV.5.5.4. Lectura del control de E/S..... 64
IV.5.5.5. Lectura..... 65
IV.5.5.6. Lectura no-destructiva..... 65
IV.5.5.7. Estatus de entrada..... 66
IV.5.5.8. Vaciar buffers de entrada..... 66
IV.5.5.9. Escritura..... 67
IV.5.5.10. Escritura con verificación..... 67
IV.5.5.11. Estatus de salida..... 67
IV.5.5.12. Vaciar buffers de salida..... 67
IV.5.5.13. Escritura del control de E/S..... 68
IV.5.5.14. Abrir dispositivo..... 68
IV.5.5.15. Cerrar dispositivo..... 68
IV.5.5.16. Medio removible..... 69
IV.5.5.17. Salida hasta que este ocupado..... 69
IV.5.5.18. IOCTL (Control de E/S)..... 69
IV.5.5.19. Toma dispositivo lógico..... 70
IV.5.5.20. Establece dispositivo lógico..... 70

IV.6. PROCESAMIENTO DE UNA PETICION TIPICA
DE ENTRADA/SALIDA..... 70

IV.7. LOS MECANISMOS PARA LA CONSTRUCCION
DE DEVICE DRIVERS..... 72

CAPITULO V
SUBSTITUCION FONETICA DIRECTA..... 76

V.1. CONCEPTOS PRELIMINARES..... 76

V.1.1. EL LENGUAJE..... 76
V.1.1.1. Lengua..... 77
V.1.1.2. Habla..... 77
V.1.1.3. Gramática..... 78
V.1.1.4. Lingüística..... 78
V.1.2. FONEMAS Y SONIDOS..... 78
V.1.2.1. Fonemas vocálicos..... 79
V.1.2.2. Fonemas consonánticos..... 79

V.1.3. LA FONOLOGIA..... 80

V.1.4. LA FONETICA..... 80

V.2. ORTOLOGIA DE LOS FONEMAS..... 81
V.2.1. CORRESPONDENCIA ENTRE FONEMAS Y LETRAS..... 82

V.3. ¿ QUE ES LA SUBSTITUCION FONETICA DIRECTA?..... 83
V.3.1. ANALISIS DE LA INFLEXION..... 83
V.3.2. ANALISIS DE LA CADENA HABLADA..... 83

CONCLUSIONES..... 84

APENDICES:

A. VOX.SYS..... 85

B. VOXTSR.COM..... 88

C. EA.EXE..... 90

BIBLIOGRAFIA..... 93

GLOSARIO..... 94

CAPITULO I

INTRODUCCION

CAPITULO I

INTRODUCCION

Este capítulo es el encargado de definir el objetivo, antecedentes, justificación y desarrollo del proyecto de evaluación final. Adicionalmente se presentan las convenciones utilizadas en este reporte escrito, así como un breve resumen de las capítulos que componen el mismo.

I.1. OBJETIVO DEL PROYECTO

El objetivo general del proyecto es el mismo que fue propuesto en el anteproyecto presentado previamente a la realización del proyecto:

Desarrollar un sistema que permita a una microcomputadora traducir a voz electrónica un texto específico en castellano, el cual puede residir en memoria principal ó secundaria.

I.2. ANTECEDENTES(DATOS PRELIMINARES)

El proyecto fue desarrollado para la empresa DIEL DE MEXICO S.A. DE C.V. Es importante mencionar que la empresa contaba con un prototipo del sistema para la conversión de frases ASCII a voz electrónica, al cual se referenciará como **"sistema de voz"** de aquí en adelante (por motivos de claridad en la redacción). El prototipo de la empresa estaba compuesto de las siguientes partes:

- Editor-analizador. Programa que procesaba una cadena de caracteres ASCII; la cadena podía provenir de la línea de comandos ó de un archivo. El análisis que se efectuaba estaba basado en la observancia de reglas para la substitución fonética directa (es decir substituir caracteres por fonemas).
- Programa TSR básico. Programa destinado a la modificación de la base de tiempo del procesador y de la transmisión de datos al diseño de hardware de la empresa.

En este contexto surgió la inquietud de desarrollar una aplicación más profesional de lo que hasta ese momento se había logrado. Entre las características deseadas para el nuevo sistema de voz se encontraban:

- Compatibilidad total con el sistema DOS y sus programas de aplicacion (procesador de palabras, lenguajes de programación, etc).
- Capacidad de modificar ó crear reglas y fonemas personalizados
- Una presentación transparente al usuario, en la cual el sistema de voz pasara a formar parte de los diferentes

dispositivos conectados a la PC.

Teniendo claro que el alcance del proyecto está referido a la mejora de un diseño prévio en el que varias cosas estaban ya definidas (reglas, fonemas, rutinas de análisis), en el presente reporte se describen los aspectos técnicos y de diseño en los que se basa el "sistema de voz".

I.3. JUSTIFICACION

La razón principal para el desarrollo de este proyecto se origina en la necesidad de dar una solución práctica cuando se trata de interfases de voz en castellano. En esta área se pueden mencionar várias aplicaciones prácticas como son: apoyo a personas no videntes (al adecuar un dispositivo de reconocimiento óptico de caracteres al sistema de voz), puntos de venta (al adecuar al sistema de voz un lector de código de barras), apoyo a los expositores (al generar aplicaciones visuales que utilicen simultáneamente el sistema de voz).

El motivo particular por el cual se desarrollo este proyecto surge cuando se detecta la necesidad de trabajar "por detrás" de las aplicaciones, es decir siendo parte del sistema operativo al aprovechar las facilidades del mismo. Otra razón se fundamenta en la necesidad de compatibilidad con los dispositivos de la PC.

I.4. DESARROLLO DEL PROYECTO

Uno de los primeros pasos en este proyecto fue la creación de un diseño innovador para el sistema de voz, el resultado de esta labor está esquematizado en la figura I-1.

Como se puede apreciar en la figura mencionada, existen varios programas en este diseño los cuales en conjunto forman el sistema de voz, la función de cada uno de ellos es analizada con detalle en los apéndices de este reporte.

El principal reto de este proyecto fue la creación de una herramienta que brindara una compatibilidad y flexibilidad tal que el usuario final del sistema de voz pudiera manipularla como si fuese un dispositivo estándar de su PC.

Partiendo de una base existente (prototipos de DIEL), se decidió aprovechar la estructura orientada a dispositivos del sistema operativo DOS, para desarrollar un nuevo programa que garantizara las características buscadas. Este programa es conocido como **"conductor de dispositivos"** (device driver), el cual representa un excelente reto para un proyecto de tesis, debido a la escasez de información fidedigna para el desarrollo de estos programas, además de la dificultad inherente de encontrar un dispositivo físico susceptible al desarrollo de un programa de este tipo.

Es importante recalcar la **dificultad de desarrollar un programa device driver**, la falta de "información confiable" es característica en esta área. Otras de las actividades del proyecto abarcaron desde la comprensión de las rutinas ya existentes, hasta

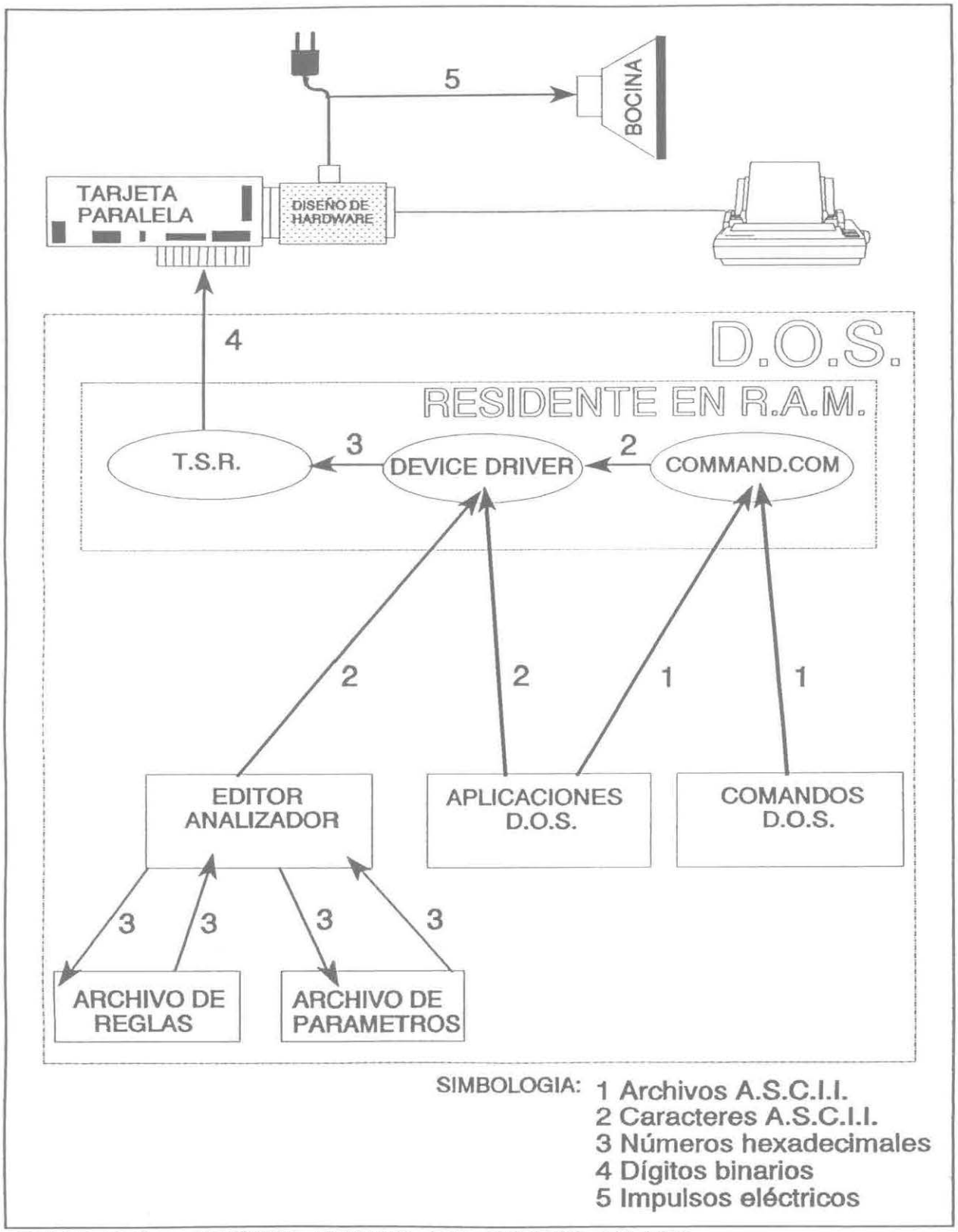


FIGURA I-1. ESQUEMA GENERAL DEL SISTEMA DE VOZ.

la adecuación de las mismas a un ambiente totalmente diferente (device drivers) en el cual las operaciones representan un alto grado de dificultad e inflexibilidad (no se puede utilizar algunas de los servicios normales del DOS, por ejmplo la INT 21h). Además figuró el establecimiento de una plataforma de comunicación confiable entre el device driver y un programa TSR.

I.5. CONVENCIONES UTILIZADAS EN ESTE REPORTE

Existen a través del reporte una variedad de términos técnicos en el idioma inglés, la presencia de estos términos obedece al hecho de que, el mismo término en castellano carece de un significado práctico (o simplemente es muy extenso), lo cual se prestaría a malas interpretaciones de la lectura. Por lo tanto, se mantienen los términos técnicos originales en la mayoría de los casos.

Con el fin de clarificar los términos antes mencionados, se incluye al final del reporte un glosario, que resulta una referencia sumamente útil cuando se duda del significado de algún término técnico.

Adicionalmente, siempre que se haga referencia a una tabla ó figura, se recomienda observarlas detenidamente, pues ellas constituyen un complemento a la información expuesta en el texto.

I.6. PANORAMA GENERAL DEL REPORTE ESCRITO

El capítulo II de este reporte nos presenta una vista general del sistema operativo DOS, su historia, sus componentes, su estructura y demás detalles interesantes de este popular sistema operativo.

En el capítulo III se ingresa más detalladamente a los procesos internos del DOS, con el fin de explicar los conceptos básicos de los manejadores de interrupciones y de los programas TSR (que son lo mismo en esencia). El capítulo continua con una descripción detallada de los diferentes tipos de programas TSR, indica cuando es adecuado implementar un programa TSR, y finaliza ofreciendo una lista de recomendaciones para su desarrollo.

El capítulo IV, el más amplio, es una valiosa guía que nos permite navegar por el confuso mundo de los conductores de dispositivos (device drivers). En este capítulo se define lo que es un dispositivo, la jerarquía de los conductores de esos dispositivos, las posibles presentaciones y tipos de los device drivers, y su estructura con gran detalle. Finalmente se da un ejemplo práctico de su utilización, así como una lista de recomendaciones similar a la del capítulo anterior.

El capítulo V, se caracteriza por las definiciones de lo que representa el lenguaje, los fonemas, y los sonidos. Para redondear estos conocimientos se efectúa un estudio de los fonemas vocálicos y se presentan las definiciones esenciales en el desarrollo de los prototipos de DIEL.

Finalmente se presenta tres apéndices, los cuales están

orientados a describir las funciones que realizan los distintos programas que componen el sistema de voz.

CAPITULO II

CONCEPTOS BASICOS DEL SISTEMA OPERATIVO (DOS)

CAPITULO II

CONCEPTOS BASICOS DEL SISTEMA OPERATIVO

(DISK OPERATIVE SYSTEM)

Para muchas personas que utilizan computadoras personales, el Sistema Operativo de Disco (DOS, por sus siglas en inglés) es la llave que abre el poderío de la máquina. Es la más visible conexión con el hardware escondido dentro del gabinete, y es a través del DOS que se pueden ejecutar aplicaciones y administrar el disco y sus archivos.

En el sentido de que abre una puerta para trabajar con una PC, el DOS es sin duda una llave, la cual entra en la familia de microprocesadores INTEL 80x86. El DOS y los chips con los que trabaja están, en realidad, conectados muy cercanamente; tan cercanos que la historia del DOS es parte de una larga historia que comprende no solo un sistema operativo sino también un microprocesador y, en retrospectiva, parte del crecimiento explosivo de la computación personal.

Seria de mucha ayuda que el lector de este escrito tenga una clara idea de lo que es el DOS antes de seguir adelante. Este capítulo esta diseñado para dar un rápido vistazo a los componentes del sistema DOS, así como una corta historia del sistema operativo.

Adicionalmente, en este capítulo se menciona la estructura e interfase del DOS y como se carga este sistema operativo dentro de la memoria de una PC.

II.1. HISTORIA DEL DOS

A través de los años el DOS a surgido como uno de los principales sistemas operativos para microcomputadoras. El DOS, sin lugar a dudas, tiene más usuarios que cualquier otro sistema operativo hoy en día. Se ha convertido en un ambiente sofisticado con herramientas y aplicaciones que satisfacen un amplio espectro de necesidades.

El DOS fue mercadeado primeramente por la compañía Seattle Computer Products (86-DOS) para su línea de computadoras. El DOS original fue escrito por Tim Paterson iniciándolo en Abril de 1980, y terminándolo en Agosto del mismo año.

Debido a que el 86-DOS trabajaba solo con procesadores 8086/8088, los cuales apenas salieron al mercado en 1980, poca gente supo de su existencia. Entonces apareció Microsoft proponiendo a Seattle Computer Products la realización de una versión "a la medida" para un cliente anónimo. A la fecha nadie sabia que IBM andaba buscando un sistema operativo. Para enero de 1981, Patterson supo quien era el cliente, y Microsoft ya había obtenido la licencia para distribuir el 86-DOS bajo su propio nombre. En Abril del mismo año, Patterson se unió a Microsoft, donde paso los siguientes meses preparando un DOS con las

especificaciones de IBM.

En Julio de 1981, Microsoft compro de Seattle Computer Products todos los derechos del 86-DOS. Cuando IBM liberó la primera PC el día 10 de Agosto de 1981, Microsoft estaba listo con el MS-DOS 1.0.

VERSIÓN	FECHA	CAMBIO EN EL HARDWARE/SOFTWARE
86-DOS	Agosto 1980	Versión de Seattle Computer Products.
1.0	Agosto 1981	PC original, disco de un solo lado.
1.1	Marzo 1982	Disco de doble lado.
2.0	Marzo 1983	PC XT, incluye disco duro.
2.1	Octubre 1983	IBM PCjr y PC portátil.
3.0	Agosto 1984	PC AT, incluye disco de alta capacidad.
3.1	Marzo 1985	Redes locales.
3.2	Diciem. 1985	Soporte realzado para nuevos medios.
3.3	Abril 1987	Soporte para PS/2.
4.0	Junio 1988	Soporte para disco duros mayores a 32Mb; integración de memoria EMS.
5.0	Mayo 1991	Mejor manejo de memoria alta, más amigable al usuario.

TABLA II-1. VERSIONES DEL DOS.

El DOS ha sufrido cambios oficiales en muchas ocasiones (y existen versiones que nunca estuvieron disponibles para el uso general). Aun y cuando las mejoras y la corrección de errores han figurado en esta evolución, cada versión generalmente esta acompañada de un cambio en el hardware. En la tabla II-1 se

describen los cambios oficiales más significativos del DOS, así como las fechas en las cuales ocurrieron esos cambios.

Mientras el DOS continua evolucionando, nuevos servicios y opciones se vuelven disponibles para los programadores. El advenimiento de ambientes gráficos (como Windows, GEM ó DESQview) ha permitido el desarrollo de nuevos servicios del DOS más sofisticados. Cada nuevo servicio esconde más a la máquina de los programas de aplicación, sin embargo, el costo de estos servicios pone en compromiso los requerimientos tradicionales de velocidad y respuesta; pero mientras los procesadores sean más rápidos, la necesidad de emplear trucos de "bajo nivel" irá disminuyendo.

II.2. COMPONENTES DEL SISTEMA DOS

El DOS, es un sistema operativo para microcomputadoras muy tradicional que consiste de cinco componentes principales:

- El cargador del sistema operativo.
- El BIOS.
- El núcleo del DOS.
- La interfase del usuario (shell).
- Programas de soporte.

Cada uno de estos componentes será discutido en las páginas siguiente.

II.2.1. EL CARGADOR DEL SISTEMA OPERATIVO

El cargador del sistema operativo es el encargado de traer a

memoria el sistema operativo desde un disco de arranque. El proceso completo de carga, llamado BOOTSTRAPPING, es bastante complejo, y múltiples cargadores pueden estar involucrados. Por ejemplo, en las implementaciones más estándar, el cargador de ROM, que es el primer programa que la computadora ejecuta cuando es encendida ó reinicializada, lee el cargador del sistema operativo del primer sector del disco de arranque y lo ejecuta. Al ejecutarse el cargador del sistema operativo, carga en memoria las principales porciones del DOS (IO.SYS y MSDOS.SYS) desde archivos convencionales de disco. Entonces el módulo especial SYSINIT (que esta dentro del MSDOS.SYS) inicializa las tablas de DOS, los buffers y se elimina a si mismo de memoria.

II.2.2. EL BIOS

El BIOS del DOS, cargado desde el archivo IO.SYS durante la inicialización del sistema, es la capa del sistema operativo que se encuentra entre el núcleo del sistema operativo y el hardware. Una aplicación realiza E/S al realizar peticiones al núcleo del sistema operativo, el cual, realiza llamadas a las rutinas del BIOS que accesan directamente al hardware. Esta división de funciones permite que las aplicaciones sean codificadas de una manera independiente del hardware.

El BIOS del DOS consiste en algún código de inicialización y una colección de device drivers (los cuales se tratan a profundidad en el capítulo IV). Los device drivers son responsables del acceso al hardware y del soporte a las interrupciones que tienen

dispositivos asociados, para que le avisen al procesador que necesitan servicio.

Los device drivers contenidos en el archivo IO.SYS, los cuales son siempre cargados durante la inicialización del sistema, son referidos algunas veces como **device drivers residentes.** A partir de la versión 2.0 del DOS, device drivers adicionales, llamados **device drivers instalables**, pueden ser cargados opcionalmente durante la inicialización del sistema mediante el comando DEVICE en el archivo CONFIG.SYS.

II.2.3. EL NÚCLEO DEL DOS

Los servicios disponibles a los programas de aplicación, por medio del DOS incluyen:

II.2.3.1. Control de procesos.

El control de las tareas (ó procesos), incluye el cargar el programa, ejecución de la tarea, terminación de la tarea, calendarización de la tarea, y comunicación entre tareas.

A pesar de que el DOS no es sistema operativo multitarea, el mismo puede tener múltiples programas residentes en memoria al mismo tiempo. Un programa puede invocar a otro, el cual se convertiría en la tarea activa. Cuando la tarea invocada termina, el programa original vuelve ha ser la tarea activa. Debido a que estos programas nunca se ejecutan simultáneamente, esta operación a manera de pila es todavía considerada como un sistema operativo de una sola tarea.

II.2.3.2. Manejo de memoria.

Debido a que la cantidad de memoria que un programa necesita varia de programa en programa, el sistema operativo tradicional provee generalmente de funciones de manejo de memoria. Los requerimientos de memoria también varían durante la ejecución de un programa, y el manejo de memoria es especialmente necesario cuando dos ó más programas están presentes en memoria al mismo tiempo.

II.2.3.3. Soporte periférico.

El sistema operativo provee soporte periférico a los programas a través de un conjunto de llamadas al sistema operativo que son traducidas por el sistema operativo en llamadas a los device drivers apropiados.

Como se establecerá posteriormente, una aplicación no necesita preocuparse de los detalles de los dispositivos periféricos ó de ninguna de las características especiales que los dispositivos pudieran tener. Debido a que el sistema operativo se encarga de todas las transferencias lógico-físicas de los dispositivos, los programas de aplicación necesitan unicamente hacer llamadas al sistema operativo.

II.2.3.4. El sistema de archivos.

El sistema de archivos es una de las porciones más largas de un sistema operativo. Un sistema de archivos es construido en el medio de almacenamiento de un dispositivo de bloques (generalmente

un disco flexible ó un disco duro) al mapear la estructura del directorio y los archivos dentro de una unidad de almacenamiento física. Un sistema de archivos en un disco contiene, como mínimo, información de localización, un directorio, y espacio para archivos.

Los sistemas de archivos son en ocasiones extendidos a mapear dispositivos de caracteres como si fuesen archivos. Estos dispositivos "archivos" pueden ser abiertos, cerrados, leidos, y escritos como si fuesen archivos normales de disco; pero todas las transferencias ocurren directamente con el dispositivo de caracteres especifico. Los "archivos dispositivos" brindan una consistencia muy útil al medio ambiente para los programas de aplicación; el DOS soporta dichos archivos al asignarles un nombre lógico reservado (como CON ó PRN) a cada dispositivo de caracteres.

II.2.4. LA INTERFASE DEL USUARIO (SHELL)

La interfase del usuario para un sistema operativo , también llamado shell ó procesador de comandos, es generalmente un programa convencional que permite al usuario interactuar con el sistema operativo. La interfase del usuario por default del DOS es un shell reemplazable llamado COMMAND.COM.

Una de las tareas fundamentales de un shell es cargar un programa en memoria al ser requerido y pasarle el control del sistema a ese programa para que así el programa se ejecute. Cuando el programa se termina, el control regresa al shell, el cual

pregunta al usuario por otro comando. Adicionalmente, el shell incluye funciones para mantenimiento de archivos y directorios. En teoría muchas de estas funciones podrían ser provistas como programas, pero hacerlas residentes en el shell le permite accesarlas rápidamente. La decisión se encuentra entre la memoria disponible contra la velocidad y flexibilidad.

II.2.5. PROGRAMAS DE SOPORTE

El DOS incluye programas de soporte que proveen acceso a facilidades del sistema operativo que no se encuentran residentes como comandos dentro del shell COMMAND.COM. Debido a que estos programas son almacenados como archivos ejecutables en disco, son esencialmente lo mismo que los **programas de aplicación** y el DOS los carga y ejecuta como si fueran cualquier otro programa.

Los programas de soporte ofrecidos con el DOS, usualmente referidos como "comandos externos", incluyen utilerías de disco como son: el FORMAT, el CHKDSK, EDLIN (que es un editor de texto orientado a líneas), y PRINT (el cual es un programa TSR que permite imprimir archivos mientras otros programas están corriendo).

II.3. ESTRUCTURA DEL DOS

A lo largo de este reporte se examinará algunas de las características más relevantes del DOS y del sistema PC, la figura II-1 puede servir de ayuda para visualizar las diferentes capas

existentes en este sistema.

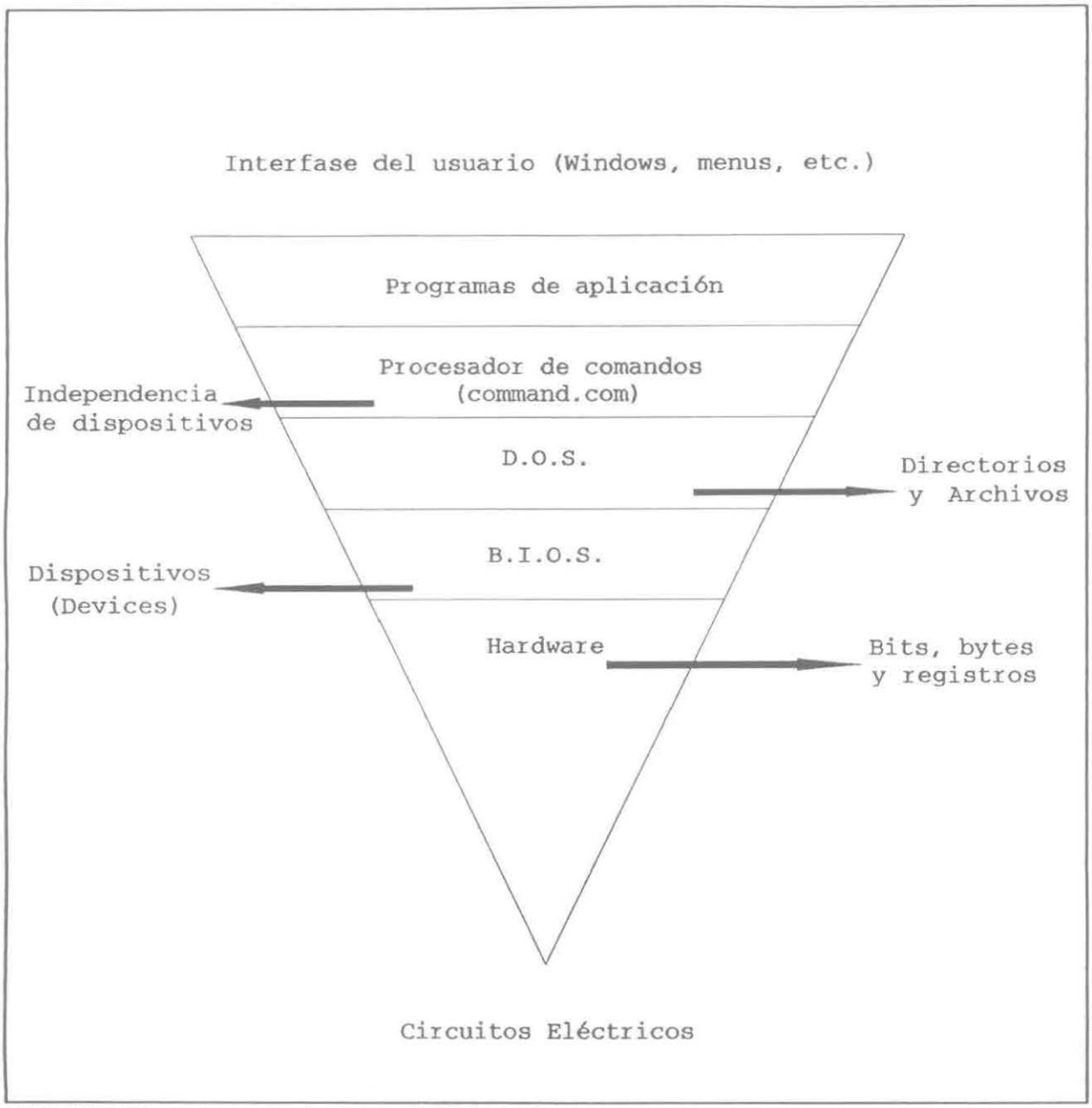


FIGURA II-1. CAPAS DE UN SISTEMA PC.

Algo fundamental en el diseño de todo el sistema DOS/PC se encuentra en el concepto de un "sistema dentro de otro sistema". Empezando con los circuitos eléctricos abajo del sistema, ellos forman un dispositivo (llamado microcomputadora) con capacidades y

características bastante precisas.

El BIOS integra este nivel de circuitos dentro de un nuevo "sistema computacional" con funciones bien definidas. El nivel del BIOS se caracteriza por la presencia de dispositivos que tienen características estándar. No importa que tipo de teclado ó monitor se tenga el BIOS hará que la mayoría respondan similarmente. Tomadas en conjunto, todas las características del BIOS hacen posible el soporte de un medio ambiente de programación. Dentro de este medio ambiente, el DOS fue desarrollado.

El medio ambiente DOS define aun otro sistema computacional, pero de un nivel mucho más alto. Un lenguaje computacional que pueda manejar archivos es introducido en este nivel, así como los dispositivos en los cuales el lenguaje esta basado. Es en este nivel donde el DOS integra las características que necesita el usuario para trabajar en un medio ambiente estandarizado. Para los programas que trabajan con DOS, los detalles de bajo nivel no son de importancia.

En el siguiente nivel (el del procesador de comandos) es todavía otra computadora; una interactiva caracterizada por su independencia de los dispositivos. A este nivel se puede trabajar con los dispositivos y los archivos como si fueran los mismos. Sin embargo DOS debe de mantener las cosas en orden, al nivel del procesador de comandos el usuario puede redireccionar la salida de datos (que se suponía que iban al monitor) a un archivo, impresora ó a voz electrónica (en el caso de utilizar el sistema de voz). En consecuencia el usuario no tienen que preocuparse de las

diferencias de manejo entre los dispositivos.

Un nivel más alto es un sistema computacional definido por los programas de aplicación del usuario. El sistema es todavía más simple cuando interactua con usuarios en vez de programadores. Ahora el "lenguaje de computación" consiste en menús, iconos y ventanas. Esta capa, como todas las demás, esta fundamentada en la capa que le precede.

II.4. INTERFASE DEL PROGRAMADOR HACIA EL DOS

Los programadores que trabajan con lenguajes de alto nivel están acostumbrados a trabajar con funciones predefinidas. En BASIC, todas las funciones (como PRINT e INPUT) están definidas en el lenguaje. Pascal también define las funciones que se pueden utilizar. Los compiladores de C vienen con un conjunto estándar de librerías de funciones. Para la mayoría de los usuarios, estas funciones representan los limites del lenguaje.

De echo, las funciones proporcionadas en BASIC, C, y Pascal están basadas en un nivel mucho más bajo de interacción con las funciones del DOS y del BIOS. Las funciones del lenguaje están codificadas para manejar peticiones de servicios en una manera estándar. Pero en orden de ser implementadas, las funciones deben referenciar directamente al DOS, al BIOS, ó al hardware.

Las interrupciones de software son el mecanismo para usar los servicios del DOS y del BIOS (el capítulo III profundiza en ellas). En el contexto actual, piense en una interrupción de software como

una llamada a una subrutina de un lenguaje de alto nivel; es necesario establecer parámetros y se obtiene un resultado.

Al ir más allá de las funciones del lenguaje, se pierden muchas cosas como: la verificación de errores, capacidad de control, y la comodidad de las librerías estándar; además se abandona una cantidad considerable de transportabilidad.

Al bajar al nivel del DOS y BIOS, se logra un alto grado de control y velocidad en los programas. El costo (un poco más de esfuerzos en la programación) es mínimo. Para lograr la velocidad más rápida posible, es necesario accesar el hardware directamente. Hacerlo solo tiene sentido cuando se desea controlar dispositivos como el monitor, el puerto serial, ó el puerto paralelo.

II.5. ¿ COMO SE CARGA EL DOS ?

El mapa de memoria del sistema DOS, como se encuentra inicialmente, se puede apreciar en la figura II-1.

Cuando ocurre una inicialización del sistema (durante el encendido de la PC), el 80x86 comienza la ejecución en la dirección FFFFh:0000h, la cual contiene el código ROM para realizar diagnósticos en el procesador, la memoria, y los dispositivos periféricos. Entonces las rutinas BIOS del ROM intentaran leer del disco de arranque. Si la lectura del disco tiene éxito, el primer sector del disco es traído a memoria y el control pasa a el **cargador del sistema operativo.**

El cargador del sistema operativo utiliza el BPB contenido en

el primer sector del disco para determinar donde se encuentra el **directorio de archivos.** El primer archivo de este directorio debe ser el archivo IO.SYS (ó el IBMBIO.COM). Si este archivo existe, es colocado en memoria y el control pasa al mismo.

FIGURA II-2. MEMORIA AL INICIALIZARSE LA PC.

La inicialización del sistema continua con la verificación de los dispositivos periféricos instalados y otros equipos; los dispositivos estándar son inicializados, los **device drivers** que son estándar para esa versión del DOS son cargados, y ciertos **vectores de interrupciones** son establecidos.

El archivo que contiene el núcleo del DOS (el MSDOS.SYS ó el IBMDOS.COM) es entonces traído a memoria. Es durante este período que el archivo CONFIG.SYS es leido para obtener los comandos especiales que definirán el medio ambiente del DOS. Entre los comandos más importantes en el proceso de inicialización se encuentran el DEVICE y el SHELL. Cada archivo que se nombre en el comando DEVICE será abierto, cargado en memoria, y enlazado en un orden predeterminado (ver capítulo IV; JERARQUIA DE LOS DEVICE DRIVERS) a los demás device drivers estándares del sistema.

Otra función que se realiza en estos instantes es la

inicialización de los nuevos device drivers para permitirles que le regresen al DOS cierta información concerniente a ellos. Si un driver es demasiado grande para caber en memoria es ignorado.

Si existe un comando SHELL en el CONFIG.SYS, este es cargado en memoria y el control pasa a el. Si no se especificó un comando SHELL, el COMMAND.COM es entonces cargado en memoria y el control pasa a el. Un esquema general de como queda distribuida la memoria después de todo este proceso, se puede apreciar en la figura II-3.

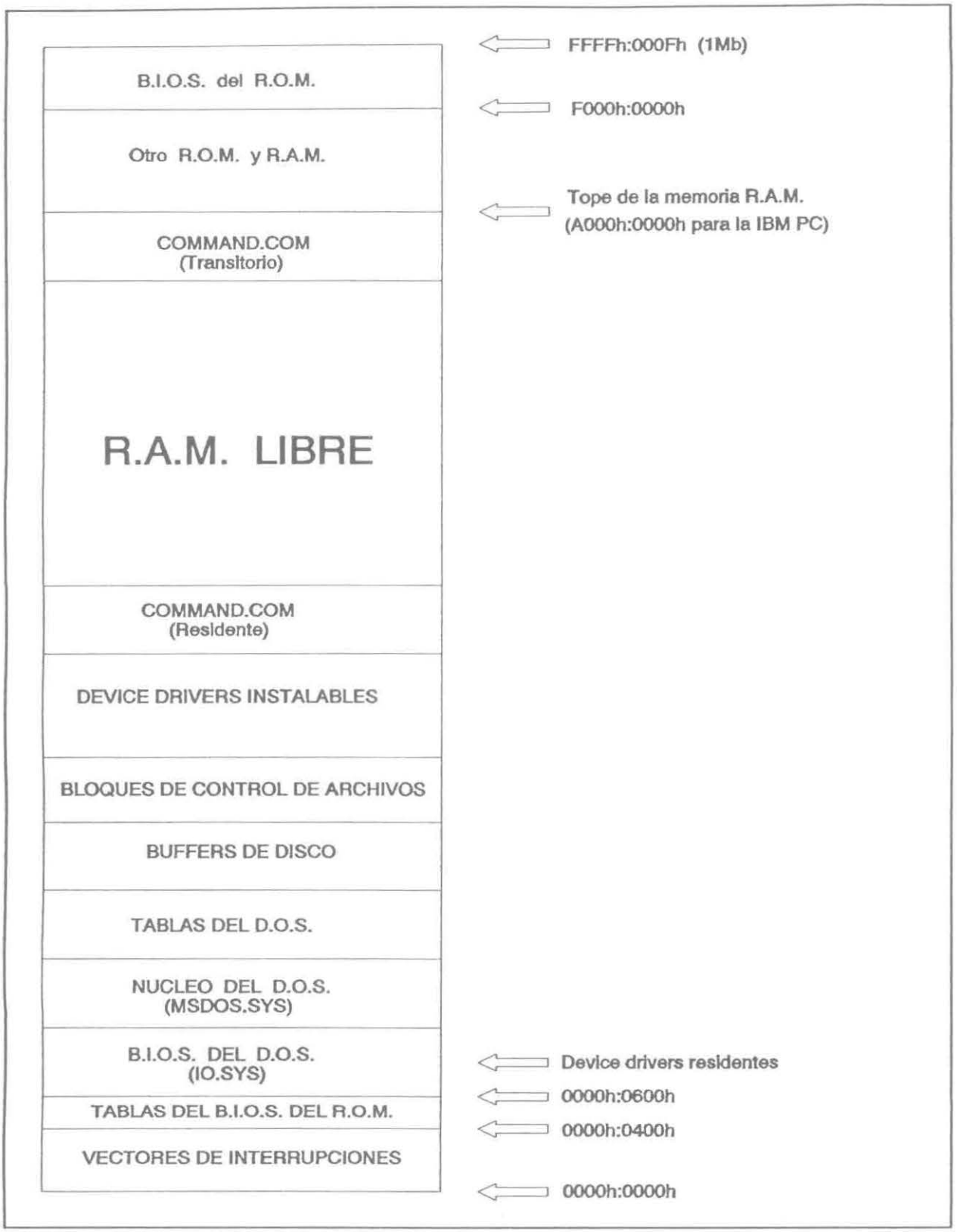


FIGURA II-3. MEMORIA DESPUES DE SER CARGADO EL SISTEMA DOS.

CAPITULO III

MANEJADORES DE INTERRUPCIONES (PROGRAMAS TSR)

CAPITULO III

MANEJADORES DE INTERRUPCIONES

(PROGRAMAS TERMINATE AND STAY RESIDENT)

En este capítulo, se realiza un vistazo profundo dentro del sistema DOS para discutir algo que tiene la reputación de ser uno de los temas más obscuros en la programación de computadoras: los manejadores de interrupciones.

Este capítulo comienza con una descripción de las interrupciones. Se discute sobre las interrupciones generadas interna ó externamente por el hardware así como las generadas por el software. Después de haber establecido los fundamentos de las interrupciones, el capítulo discute los pormenores de los programas TSR, que son en realidad manejadores de interrupciones, pues cada TSR se enlaza de alguna manera a alguna interrupción y responde a ella como lo hace una **Rutina de Servicio a Interrupciones (ISR)**. Este capítulo describe las características de un TSR y recomienda los lugares más apropiados donde utilizar un programa TSR.

III.1. ¿ QUE ES UNA INTERRUPCION ?

Una interrupción es una señal dirigida al procesador que

indica que ha ocurrido un evento que necesita atención especial. Es utilizada para captar la atención del procesador hacia algo importante. Si no existieran las interrupciones, habría que verificar el estado de todos los dispositivos del sistema para saber si a alguno de ellos necesita algún servicio. Esta es la razón por la cual se inventaron las interrupciones: para eliminar la necesidad de verificar cada dispositivo y evitar en consecuencia un degradamiento en el tiempo de respuesta a eventos externos.

En cualquier momento que la computadora detecta una condición de interrupción, el procesador salva lo que este haciendo en ese momento, "marca el lugar" en el programa, y transfiere el control a un **manejador de interrupciones (handler)**, del que se espera que regrese después de haber dado servicio a la interrupción.

Las interrupciones son manejadas muy eficientemente en la familia de procesadores 80x86, por medio de múltiples tipos de interrupciones ó niveles. Cada tipo tiene una dirección reservada en memoria, llamada **el vector de interrupción**, que especifica donde se encuentra localizado el programa que manipulará una interrupción en particular. Este diseño permite un procesamiento veloz de la interrupción debido a que el procesador puede transferir el control directamente a la rutina apropiada; es decir, no necesita de una rutina central que desperdicie valiosos ciclos de máquina determinando la causa de la interrupción. Este concepto de tipos de interrupciones permite que las interrupciones sean priorizadas, así si varias interrupciones son generadas simultáneamente, la más importante será procesada primero.

III.2. TIPOS DE INTERRUPCIONES

La familia de microprocesadores INTEL 80x86 soporta 256 niveles de interrupciones priorizadas, las cuales pueden ser accionadas por tres tipos de eventos: interrupciones internas del hardware, interrupciones externas del hardware, e interrupciones de Software; las cuales se discutirán a continuación con más detalle.

III.2.1. INTERRUPCIONES INTERNAS DEL HARDWARE

En ocasiones estas interrupciones también son llamadas "fallas", son generadas por ciertos eventos encontrados durante la ejecución de algún programa, por ejemplo un intento de realizar una división por cero. La asignación de tales eventos a ciertas interrupciones esta conectada dentro del procesador y **no son modificables**. La tabla III-1 presenta una lista de las interrupciones internas de la familia 80x86.

III.2.2. INTERRUPCIONES EXTERNAS DEL HARDWARE

Estas interrupciones son accionadas por controladores de dispositivos periféricos ó por coprocesadores como el 8087 ó 80287. Ellos pueden estar ligados a el pin de interrupción no-enmascarable (NMI) ó a su pin de interrupción enmascarable (INTR). La línea NMI es normalmente reservada para interrupciones causadas por eventos catastróficos como un error de paridad en la memoria ó fallas de corriente.

En vez de estar conectados directamente al procesador, las interrupciones de dispositivos externos pueden ser canalizadas a

NIVEL DE LA INTERRUPCION	DIRECCION DEL VECTOR	SIGNIFICADO
(INTERRUPCIONES DE HARDWARE DEL 8086)		
00h	00h	División por cero.
01h	04h	Paso unitario.
02h	08h	Interrupción no enmascarable.
03h	0Ch	Punto de rompimiento.
04h	10h	Overflow.
(INTERRUPCIONES DE HARDWARE DEL 80286)		
05h	14h	Exceso en los limites del rango.
06h	18h	Código inválido
07h	1Ch	Extensión del procesador no disponible.
08h	20h	Excepción doble.
09h	24h	Sobrepaso del segmento.
0Ah	28h	Segmento de tarea inválido.
0Bh	2Ch	Segmento no presente.
0Ch	30h	Sobrepaso del segmento stack.
0Dh	34h	Falla de protección general.
(INTERRUPCIONES DE HARDWARE DEL 80386)		
0Eh	38h	Falla de página.
0Fh	3Ch	Reservada.
10h	40h	Error del coprocesador numérico.
11h-1Fh	44h	Reservadas.

TABLA III-1. INTERRUPCIONES INTERNAS DEL HARDWARE.

través de un dispositivo llamado el **controlador programable de interrupciones (PIC) INTEL 8259A.** El procesador controla el PIC a través de un conjunto de puertos de E/S, y el PIC, a su vez, envía señales al procesador a través de el pin INTR. El PIC permite que las interrupciones de dispositivos específicos puedan ser activadas y desactivadas, ó ajustar sus prioridades, bajo el control de un programa.

Cada PIC puede manejar solamente ocho niveles de interrupciones. Sin embargo, los PIC pueden ser configurados en conjunto en una estructura de árbol, para así obtener tantos niveles como se desee. Adicionalmente las interrupciones INTR pueden ser activadas ó desactivadas globalmente por intermedio de las instrucciones STI y CLI del procesador. Como es de esperarse estas instrucciones no tienen efecto en las interrupciones recibidas a través del pin NMI.

Los fabricantes de sistemas computacionales y/o los fabricantes de los dispositivos periféricos asignan dispositivos externos a niveles de interrupción PIC específicos. Estas asignaciones son realizadas como conexiones eléctricas físicas y **no pueden ser modificadas** por el software. Sin embargo **si se puede manipular** la manera en como trabajan estos dispositivos, un ejemplo claro de esta manipulación se tratara en el apartado PROGRAMAS TSR HIBRIDOS.

III.2.3. INTERRUPCIONES DE SOFTWARE

Cualquier programa puede accionar las interrupciones de software sincrónicamente, simplemente ejecutando la instrucción INT. El DOS utiliza las interrupciones de la 20h a la 3Fh para comunicarse con sus módulos y con los programas de aplicación (Por ejemplo, el ejecutador de DOS arranca al realizar una interrupción 21h). El ROM BIOS de la PC y el software de aplicación utiliza otras interrupciones, con números más altos ó bajos, para varios propósitos, ver tabla III-2.

VECTOR	ACCION
00h	División por cero.
01h	Paso único.
02h	Interrupción no enmascarable.
03h	Punto de ruptura.
04h	Overflow.
05h	Imprimir pantalla.
06h	No usado.
07h	No usado.
08h	IRQ 0 de hardware (pulso del reloj).
09h	Interrupción de entrada del teclado.
0Ah	Reservado.
0Bh	Controlador del puerto asincrónico 1 (COM2).
0Ch	Controlador del puerto asincrónico 0 (COM1).
0Dh	Controlador de disco duro.
0Eh	Controlador de disco flexible.
0Fh	Controlador de impresora.
10h	Driver de video.
11h	Verificación de la configuración del equipo.
12h	Verificación de el tamaño de la memoria.
13h	Disco flexible ó duro (PC/XT).
14h	Driver del puerto de comunicaciones.
15h	Servicio de cassette/red.
16h	Driver del teclado.
17h	Driver de la impresora.
18h	BASIC en ROM.
19h	Reinicializador del sistema.
1Ah	Lee/establece el reloj real.
1Bh	Manejador del Control-Break.
1Ch	Pulso del reloj (definido por el usuario).
1Dh	Tabla de parámetros de video.
1Eh	Tabla de parámetros de disco.
1Fh	Tabla de caracteres gráficos.
20h	Terminación de programa.
21h	Despachador de funciones del DOS.
22h	Vector de terminación.
23h	Vector del Control-C.
24H	Vector de error critico.
25h	Lectura absoluta a disco.
26h	Escritura absoluta a disco.
27h	Terminar y quedar residente (TSR).
28h	Interrupción de DOS OK.
2Fh	Interrupción Multiplex.
40h	Driver de disco (PC/XT).
41h	Tabla de parámetros de disco duro.
43h	Tablas de caracteres gráficos.

TABLA III-2. PRINCIPALES INTERRUPCIONES DE SOFTWARE.

Estas asignaciones a las interrupciones son simplemente convenciones y no están conectadas al hardware de ninguna manera, por lo tanto **si pueden ser modificadas** ó substituidas.

III.3. LA TABLA DEL VECTOR DE INTERRUPCIONES

A los 1024 bytes más bajos de la memoria del sistema, se les conoce como la **tabla del vector de interrupciones**, cada vector ocupa 4 bytes, dando un total de 256 vectores de interrupción distintos.

Cada vector de interrupciones es un apuntador FAR (32 bits, en la forma desplazamiento:segmento) a la rutina manejadora de la interrupción actual; ó, como en muchos casos, el vector puede contener la dirección de una tabla de datos (por ejemplo la tabla de los caracteres gráficos apuntada por la Interrupción 1Fh).

Las interrupciones de la 00h a la 1Fh (las de bajo nivel) son utilizadas por las interrupciones internas del hardware; el DOS utiliza las interrupciones que van de la 20h a la 3Fh; y las demás interrupciones están disponibles para el uso de dispositivos de hardware externos, drivers del sistema y software de aplicación.

III.4. COMO TRABAJAN LAS INTERRUPCIONES

Cuando una interrupción ocurre, el procesador puede estar en cualquier estado. Los procesadores están diseñados de tal manera que siempre completan cualquier instrucción en proceso, antes de

responder a una interrupción. Cuando el procesador reconoce una interrupción, responde guardando en el STACK: el registro de las banderas (PSW), el apuntador a las instrucciones (IP), y el registro del segmento de código (CS). Acto seguido desabilita a las demás interrupciones.

Después de que esta información crítica sobre el estado de la máquina a sido guardada, el procesador busca en el bus del sistema un número de 8 bits, que es **el nivel de la petición de interrupción (IRQ).** Este nivel identifica exactamente cual dispositivo a realizado la interrupción y le permite al procesador saber cual vector (manejador de interrupción) utilizar en respuesta.

En el ambiente PC (y sus sucesores), un desplazamiento fijo de 8 se le suma al IRQ, para determinar que interrupción es señalada. Por ejemplo un IRQ de nivel 0 genera la Interrupción 08h, y la IRQ de nivel 7 genera la Interrupción 0Fh.

Acto seguido el procesador multiplica el número de interrupción por 4 para obtener el desplazamiento dentro de **la tabla del vector de interrupciones**, y entonces va al segmento 0000h para encontrar el vector apropiado. El contenido del vector se coloca en el CS:IP, y el control del sistema se transfiere automáticamente a la primera instrucción del programa que procesa la interrupción **(el manejador de interrupciones).** Un resumen de la secuencia de operaciones se puede observar en la figura III-1.

Cuando el procesador se encuentra en el manejador de interrupciones, el manejador controla al procesador. La mayoría de los manejadores primero habilitan las interrupciones, en orden de

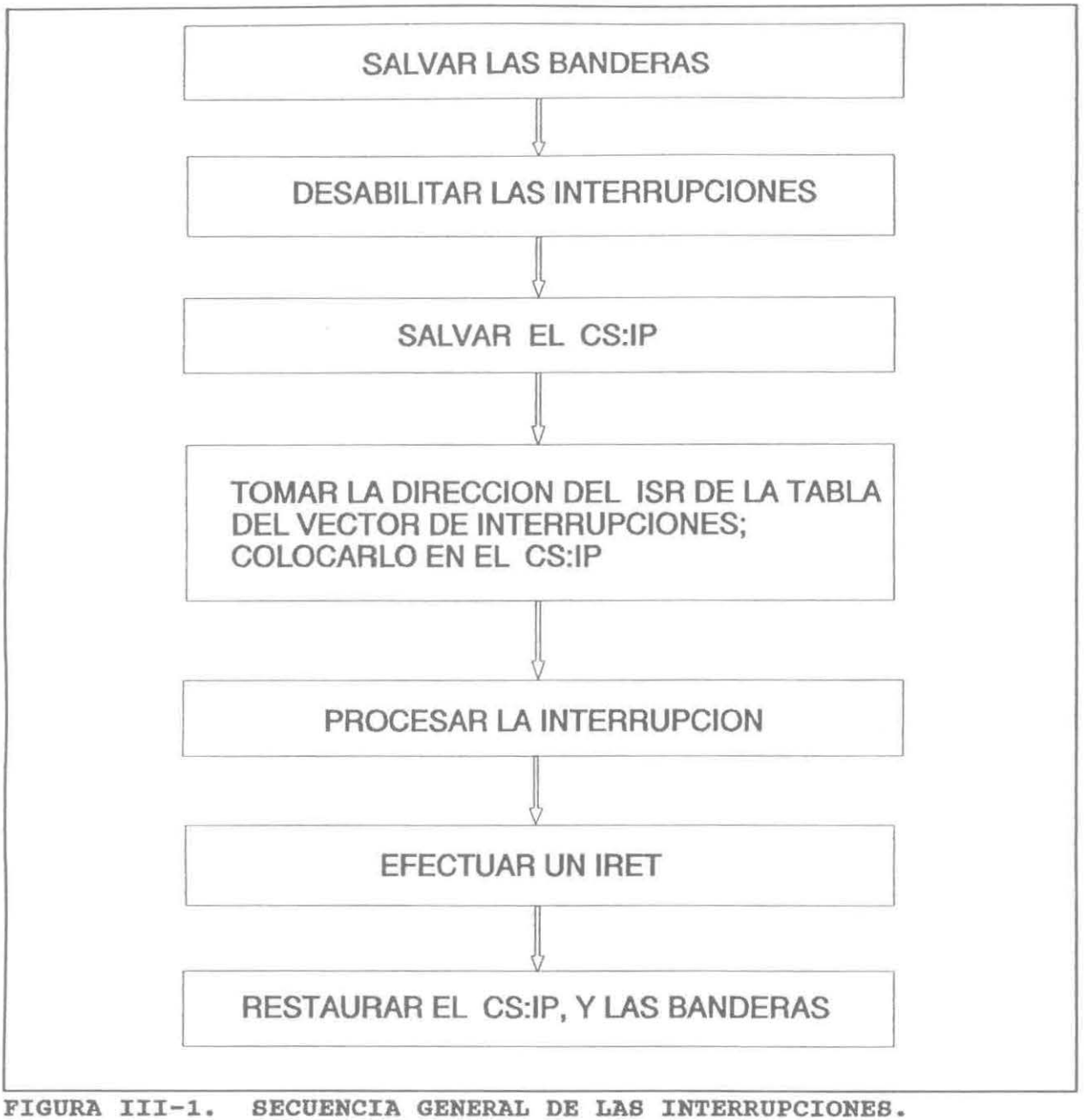


FIGURA III-1. SECUENCIA GENERAL DE LAS INTERRUPCIONES.

que las interrupciones de alta prioridad puedan ser atendidas. Los manejadores también salvan otros registros que ellos utilizan y realizan sus operaciones lo más pronto posible.

III.5. ¿ QUE ES UN PROGRAMA TSR ?

Los programas que se **terminan y quedan residentes en memoria (TSR)**, originalmente fueron pensados como una manera muy conveniente de agregar **rutinas de servicio a interrupciones (ISR)** al DOS, al permitir que algunos programas se inicializen a si mismos y se enlacen dentro de la estructura del sistema de interrupciones.

Los programas TSR son un tema siempre de actualidad en la programación de las PC. La introducción de el programa SIDEKICK de la compañía BORLAND causo una explosión de aplicaciones TSR dentro de el mercado de software para PC. La gran mayoría de estas aplicaciones TSR, no cumplen con el propósito original de la función TSR, el cual es ser una extensión del sistema operativo.

El concepto de los TSR a sufrido algunos cambios significativos desde que el DOS fue introducido. La función TSR original (Interrupción 27h) a sido reemplazada por la Interrupción 21h, Función 31h. Esta función es más conveniente de utilizar porque permite pasar un código de regreso y permite al TSR que disponga de más de 64Kb de memoria.

III.6. ESTRUCTURA DE UN PROGRAMA TSR

La llamada TSR (Interrupción 21h, Función 31h y Interrupción 27h) del sistema DOS, permite al programador instalar código ejecutable ó datos de un programa en un bloque reservado del RAM, donde este reside mientras otros programas se ejecutan. Datos

globales, manejadores de interrupciones, y aplicaciones enteras pueden estar residentes en RAM de esta manera.

El código ejecutable y los datos de un TSR pueden separarse en dos porciones: una **residente en RAM** y otra **transitoria** (ver figura III-2). La porción residente en RAM del TSR contiene código ejecutable y datos para una aplicación que realiza alguna función útil en demanda. La porción transitoria instala al TSR; esto lo hace en el siguiente orden:

FIGURA III-2. ORGANIZACION DE UN PROGRAMA TSR EN MEMORIA.

- Inicializa los datos y los manejadores de interrupciones contenidos en la porción residente en RAM.
- Ejecuta la llamada para TSR que deja la porción residente en RAM fija en memoria.
- Libera la memoria utilizada por la porción transitoria.

El código de la porción transitoria del TSR corre cuando se llama a ejecución al archivo .EXE ó .COM que contiene al programa TSR; el código de la porción residente en RAM corre solamente cuando es invocado explícitamente por un programa controlador ó por la ejecución de una interrupción de hardware ó de software.

El DOS provee de las facilidades listadas en la tabla III-3, para instalar manejadores de interrupciones en una manera que no

FUNCIÓN	ACCION
INT 21h FUNCIÓN 25h	Actualiza el contenido de un vector de interrupciones.
INT 21h FUNCIÓN 31h	Activa TSR preferido sobre la INT 27h.
INT 21h FUNCIÓN 34h	Toma la dirección de bandera InDOS, la cual determina si es apropiado entrar a un programa TSR.
INT 21h FUNCIÓN 35h	Toma la dirección de un vector de interrupciones.
INT 27h	Activa TSR, disponible por compatibilidad con DOS 1.x.
INT 2Fh	Detección estándar de la presencia de algún TSR.

TABLA III-3. FACILIDADES QUE BRINDA DOS A LOS PROGRAMAS TSR.

interfiere con las funciones del sistema operativo ó de otros manejadores de interrupciones.

Estas funciones permiten al programador examinar ó modificar el contenido de la tabla del vector de interrupciones y además reservan memoria para el uso de un manejador sin causar conflictos de memoria.

III.7. TIPOS DE PROGRAMAS TSR

Dependiendo del método por el cual el control se le transfiera al programa residente en RAM, los programas TSR generalmente se pueden clasificar dentro de tres grupos:

III.7.1. PROGRAMAS TSR ACTIVOS

Son el tipo más común de TSR (por ejemplo SIDEKICK), generalmente se activan como respuesta a una combinación predeterminada de teclas llamada "hotkey". Cuando estos programas se activan, toman el control de la computadora y realizan su función antes de regresar el control a el programa que originalmente controlaba a la PC.

III.7.2. PROGRAMAS TSR PASIVOS

Estos programas TSR responden cuando un programa controlador llama a una interrupción ó los activa explícitamente. Cuando son llamados, realizan una función definida, similar a una subrutina, y entonces regresan al programa controlador.

Los TSR inactivos trabajan en un ambiente más benigno, pues el programa que los llama no es interrumpido por el TSR, así que el estatus del DOS, el BIOS, y el hardware están bien definidos cuando el programa TSR comienza su ejecución.

III.7.3. PROGRAMAS TSR HIBRIDOS

Para estos programas TSR la activación ocurre cuando un programa controlador los activa alterando directamente su vector de interrupciones. Cuando son llamados, alteran ó modifican el funcionamiento normal de un dispositivo, y entonces regresan al programa controlador; de ese momento en adelante el dispositivo presentará características diferentes a las normales.

Un ejemplo claro de estos TSR es el TSRVOX el cual en primera

instancia es activado desde el device driver VOX explícitamente, y de allí en adelante se activa respondiendo a un numero específico de pulsaciones de reloj (IRQ 0, Interrupción 08h timer tick), las cuales obedecen a una base de tiempo modificada por el propio TSR.

III.8. ¿ CUANDO UTILIZAR UN PROGRAMA TSR ?

Existen muchas razones válidas para codificar un programa TSR (ó manejador de interrupciones), entre las principales se encuentran:

- Para reemplazar el manejador por default de DOS para una interrupción interna de hardware (por ejemplo: divisiones por cero, exceso de limites, etc).
- Para reemplazar el manejador por default de DOS para una excepción del sistema predefinida (por ejemplo: el manejador de errores críticos, ó el manejador del Control-C).
- Para dar servicio a interrupciones no soportadas por los device drivers estándar del DOS (por ejemplo: el puerto serial de comunicaciones, el cual puede funcionar a velocidades más altas que las normales).
- Para proveer un camino de comunicación entre un programa que termina y queda residente, y otro software de aplicación (por ejemplo: la rutinas GRAB de WORD PERFECT la cual captura el contenido de la pantalla gráfica y la redirecciona a un archivo predeterminado).
- Para encadenar un manejador de interrupciones del usuario

dentro del manejador por default del sistema DOS para un dispositivo de hardware, permitiendo que las acciones del usuario y las del sistema DOS ocurran en esa interrupción.

Con respecto al último argumento, en el caso particular del sistema de voz , el programa TSRVOX.COM se "cuelga" de la interrupción 08h (Pulso de tiempo de la PC) y "reprograma" al reloj con una base de tiempo adecuada para la transmisión de dígitos binarios a un diseño de hardware. Esto se realiza así, porque la base de tiempo normal de una PC es de 18.2 veces por segundo, la cual es una base de tiempo muy pobre para los propósitos del proyecto (ocasiona que la voz se escuche con un tono no muy adecuado). Al reprogramar el reloj para que la base de tiempo sea más dinámica, se permiten tres cosas:

- Se reduce los ciclos de espera inútiles del procesador al poder realizar una transmisión de datos oportuna.
- Se simula una capacidad de multitarea entre el sistema de voz y los demás eventos de la PC.
- La voz electrónica resultante tiene un sonido más adecuado.

III.9. LOS MECANISMOS PARA LA CONSTRUCCION DE UN PROGRAMA TSR

Los manejadores de interrupciones generalmente deben de estar codificados de una manera reducida y veloz. La mayoría de ellos se programan en el lenguaje Assembler para eliminar cargas excesivas y para asegurar que la rutina se ejecuta lo más rápido posible. Se puede programar manejadores en C, pero se sugiere que las

interrupciones de tiempo críticas sean manejadas con la mínima carga posible. En la tabla III-4 se puede observar cuales son las herramientas necesarias para programar un programa TSR.

HERRAMIENTA	DESCRIPCIÓN
Editor	Un procesador de palabras o un editor de texto el cual permita incluir texto fuente a un archivo. Es también utilizado para modificar un **archivo de texto.**
Assembler	Un programa que convierta un programa fuente, escrito en lenguaje ensamblador, a **módulos objeto relocalizables.**
LINK	Un programa enlazador el cual combina uno o mas **módulos objeto relocalizables** dentro de un **archivo ejecutable.**
EXE2BIN	Un programa que convierte un **archivo ejecutable** en **archivos imagen de memoria.** Estos archivos son conocidos como archivos ".COM" y son indispensables para permitir que un programa resida en memoria.

TABLA III-4. HERRAMIENTAS MINIMAS PARA DESARROLLAR UN PROGRAMA ".COM".

Cuando se codifique un manejador de interrupciones, es recomendable no utilizar las funciones del DOS a menos que se tome especial cuidado. El **DOS no es reentrante**, y si es interrumpido cuando esta haciendo algo, se pude caer fácilmente en un bloqueo del sistema al llamar a las funciones del DOS.

No importa como se agrege un TSR al sistema, el TSR debe de reconocer la posible presencia de otros TSR en el sistema. Para permitir que otras operaciones se realicen en la activación, el TSR debe de llamar primero a la rutina original que manejaba la

interrupción antes de comenzar con las actividades del TSR.

A manera de conclusión en la tabla III-5 se presentan una serie de recomendaciones que deben de tomarse en cuenta si se desea desarrollar programas TSR.

1. Nunca llamar a funciones del DOS, si necesita realizar alguna operación de E/S pude hacer lo siguiente: - Utilizar alguna función del BIOS. - Monitorear la bandera InDOS. Cuando esta bandera no es cero, DOS esta ejecutando alguna interrupción 21h. - Monitorear la Interrupción 28h. Esta interrupción informa si el DOS esta esperando el resultado de una interrupción 21h.
2. Crear una función que le permita indicar al TSR si esta presente, esto se puede hacer mediante una firma en memoria.
3. Encadenar siempre cualquier interrupción usada por el TSR, esto se hace pasando el control al vector de interrupciones que el TSR encontró cuando inicio su trabajo.
4. Utilizar un stack local en vez de uno controlado por un programa de interrupciones.
5. Se recomienda la utilización de la interrupción 21h función 25h (Set Interrupt Vector) para modificar el vector de interrupciones; no escriba directamente en la tabla del vector de interrupciones.

TABLA III-5. CONSIDERACIONES AL DESARROLLAR UN PROGRAMA TSR.

CAPITULO IV

CONDUCTORES DE DISPOSITIVOS (DEVICE DRIVERS)

CAPITULO IV

CONDUCTORES DE DISPOSITIVOS

(DEVICE DRIVERS)

El software que se utiliza en los sistemas computacionales modernos está, por convención, organizado en capas con diferente grado de independencia con respecto al hardware de la computadora. Los propósitos de esta organización son básicamente:

- Para minimizar el impacto en los programas con respecto a diferencias entre dispositivos de hardware ó cambios en el hardware.
- Para permitir que el código de operaciones comunes sea centralizado y optimizado.
- Para facilitar la tarea de mover programas y sus datos de un lugar a otro.

La capa **superior** y más independiente del hardware es usualmente la transitoria, ó de programas de aplicación, la cual ejecuta una tarea específica y trata con datos en términos de archivos ó registros dentro de esos archivos. Estos programas se llaman transitorios porque son traídos al RAM cuando son necesitados y son descartados de la memoria cuando su trabajo a terminado.

La capa **intermedia** es el núcleo del sistema operativo, el cual administra la localización de los recursos del sistema como son la memoria y el almacenamiento en disco, además provee una variedad de servicios a los programas de aplicación.

Los módulos en la capa **inferior** se llaman device drivers. Estos device drivers son los componentes del sistema operativo que administra el controlador, ó adaptador, de un dispositivo periférico; que es una pieza de hardware que la computadora utiliza con propósitos como: el almacenamiento ó comunicación con el mundo exterior. Por lo tanto, los device drivers son responsables de la transferencia de datos entre el dispositivo periférico y la memoria RAM de la computadora, donde otros programas podrán trabajar con ellos. Los device drivers protegen al núcleo del sistema operativo de la necesidad de tratar con direcciones de puertos de E/S, características operativas, y las peculiaridades de un dispositivo periférico en particular. De la misma manera el núcleo del sistema operativo protege a los programas de aplicación de los detalles del manejo de archivos.

Una abstracción conceptual de lo expuesto hasta el momento se puede apreciar en la figura IV-1, en la cual todo gira alrededor del núcleo del DOS.

A partir de la versión 2.0 del DOS, los device drivers se separaron del núcleo del sistema operativo. Esto se debio a los diferentes métodos empleados por cada fabricante para el manejo de los dispositivos periféricos existentes en el mercado.

Los device drivers tienen una estructura confiable y se

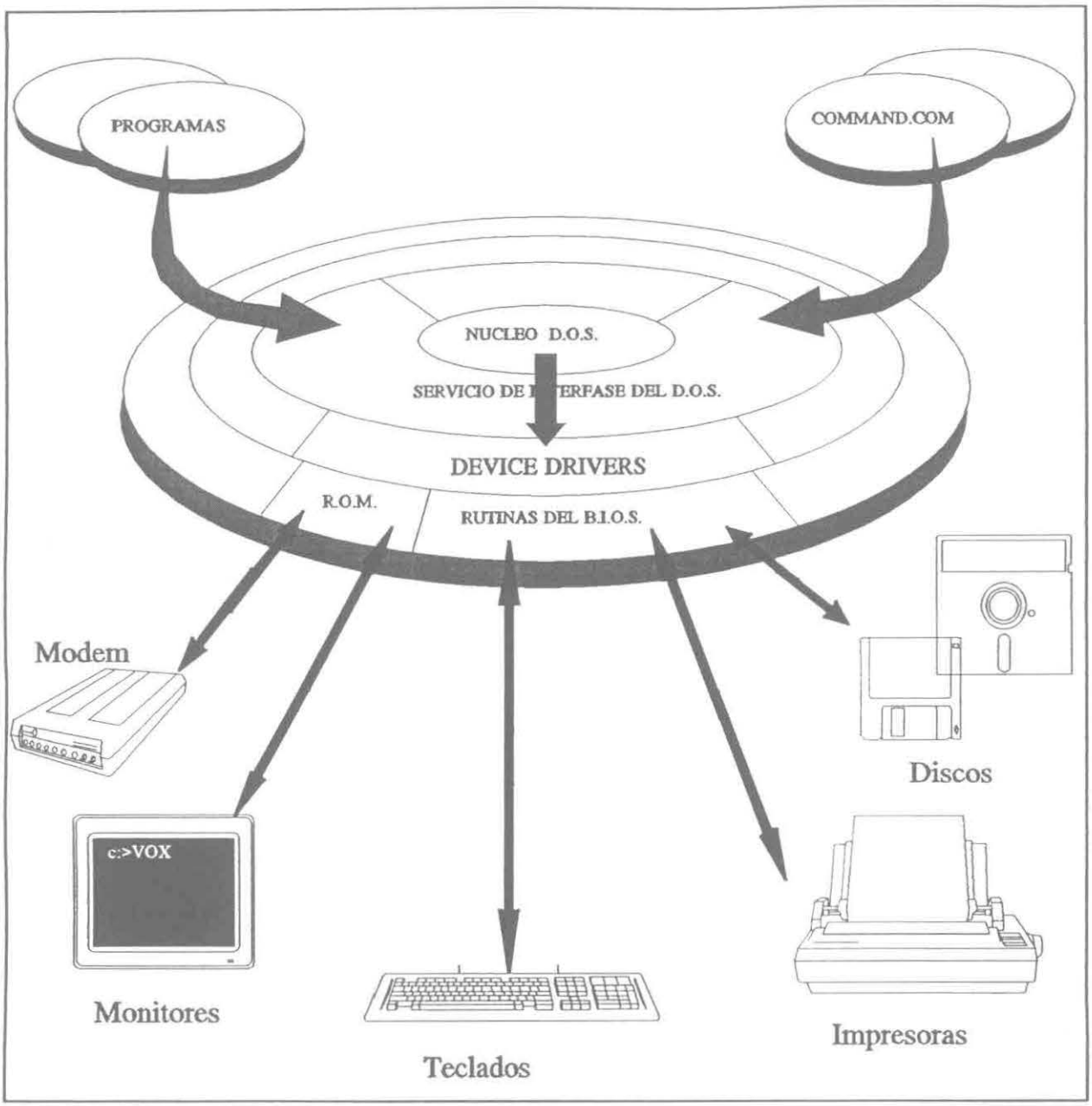


FIGURA IV-1. UN MODELO CONCEPTUAL DEL DOS.

comunican con el núcleo del sistema operativo a través de un esquema sencillo y claramente definido, el cual consiste de llamadas FAR, códigos de operación, y paquetes de datos. Obteniendo la información adecuada con respecto al hardware, un programador puede escribir un nuevo device driver prácticamente

para cualquier dispositivo periférico.

Este capítulo explica en detalle la anatomía, operación, y creación de device drivers para el DOS versiones 3.0 y posteriores.

IV.1. DISPOSITIVOS DE LAS COMPUTADORAS PERSONALES

Los dispositivos de una PC son las piezas de hardware que se le pueden "colgar" a la máquina. Una lista de los posibles tipos de dispositivos existentes así como su tipo de interacción con la PC, se encuentran en la tabla IV-1.

TIPO DE INTERACCION	POSIBLES DISPOSITIVOS
Entrada:	Digitalizador de imágenes. CD-ROM. Lectora óptica de caracteres. Lectora de código de barras. Digitalizador de gráficas. Mouse. Track-ball. Convertidor Análogo-Digital.
Entrada/Salida:	Redes locales. Drives de respaldo en cinta. Grabadora de video-cassette. Drives de disco duro ó diskette.
Salida:	Plotter. Programador de chips ROM. Impresoras láser. Sintetizadores de sonido. Convertidor Digital/Análogo.

TABLA IV-1. POSIBLES DISPOSITIVOS QUE SE PUEDEN AÑADIR A UNA PC.

El DOS permite a los programas de aplicación controlar un conjunto de dispositivos estándar: teclado, pantalla, discos, y

NOMBRE DEL DISPOSITIVO D.O.S.	DISPOSITIVO ESTANDAR
con:	Teclado / Pantalla
com1:	Puerto serial #1
aux:	Puerto auxiliar (idéntico a com1:)
com2:	Puerto serial #2
lpt1:	Puerto paralelo #1
lpt2:	Puerto paralelo #2
lpt3:	Puerto paralelo #3
prn:	Puerto lógico de impresora (idéntico a lpt1:)
nul:	Dispositivo nulo
clock$	Reloj de Software
A:	Primera unidad de diskette
B:	Segunda unidad de diskette
C:	Disco duro (normalmente)

TABLA IV-2. NOMBRES ESTANDAR QUE DOS ASIGNA A SUS DISPOSITIVOS.

puertos paralelos y seriales. Cada dispositivo del DOS tiene un nombre único asociado, y es a través de estos nombres que los programas de aplicación pueden accesar los dispositivos. En la tabla IV-2 se listan los nombres de los dispositivos estándar del DOS, como han sido definidos a partir de la versión 2.0.Cada sistema DOS contiene drivers internos para la consola (combinación de teclado y monitor), el puerto serial, el puerto paralelo, el reloj interno, y al menos un dispositivo de almacenamiento en disco

(el dispositivo de arranque del sistema). Estos drivers, conocidos como los **device drivers residentes ó estándar**, son cargados en conjunto desde el archivo IO.SYS cuando el sistema es prendido ó inicializado.

Los drivers para dispositivos periféricos adicionales se encuentran en archivos separados en los discos del fabricante. Estos drivers, llamados **device drivers instalables**, son cargados y enlazados dentro del sistema durante la inicialización del mismo, como resultado de las directivas " DEVICE = " que se encuentren en el archivo CONFIG.SYS. Ejemplos de estos drivers son los archivos ANSI.SYS y EGA.SYS incluidos con cada versión del DOS. En todos los aspectos, los device drivers instalables tienen la misma estructura y relación con el núcleo del sistema operativo, que la que tienen los residentes.

IV.2. JERARQUIA DE LOS DEVICE DRIVERS

El DOS administra las peticiones de acceso a los dispositivos, generando códigos de comando al device driver apropiado. Cada device driver contiene el nombre del dispositivo específico que está controlando, y DOS localiza el device driver apropiado buscando a través de una lista de device drivers residentes e instalados.

DOS mantiene una lista enlazada de los drivers comenzando con el dispositivo nulo (NUL). El device driver para NUL es el primero de la lista y contiene un apuntador al siguiente device driver. A su vez, cada device driver apunta al siguiente. El apuntador para

el último device driver va a contener el valor -1, de esta manera se señala el fin de la lista.

La administración de los drivers residentes e instalables, es realizado por DOS mediante un relativamente simple mecanismo. Como se puede apreciar en la figura IV-2, la lista de los dispositivos residentes de DOS comienza con NUL y le siguen CON, AUX, PRN, etc. La lista de los device drivers de DOS es llamada **la cadena de dispositivos**, la cual es una lista enlazada de los programas device drivers actuales. Estos programas device drivers residen en un área de la memoria de la PC que DOS utiliza. Siempre que un nuevo driver es instalado, DOS lo inserta en la lista justo después del dispositivo NUL. Esto permite al usuario reemplazar un device driver residente, debido a que cualquier petición de dispositivos ocasionará que DOS busque en esta lista comenzando por NUL. Si el usuario reemplaza un device driver residente por uno de su propiedad, el DOS encontrará primero al nuevo device driver y nunca llegará a alcanzar al device driver original del mismo nombre. De

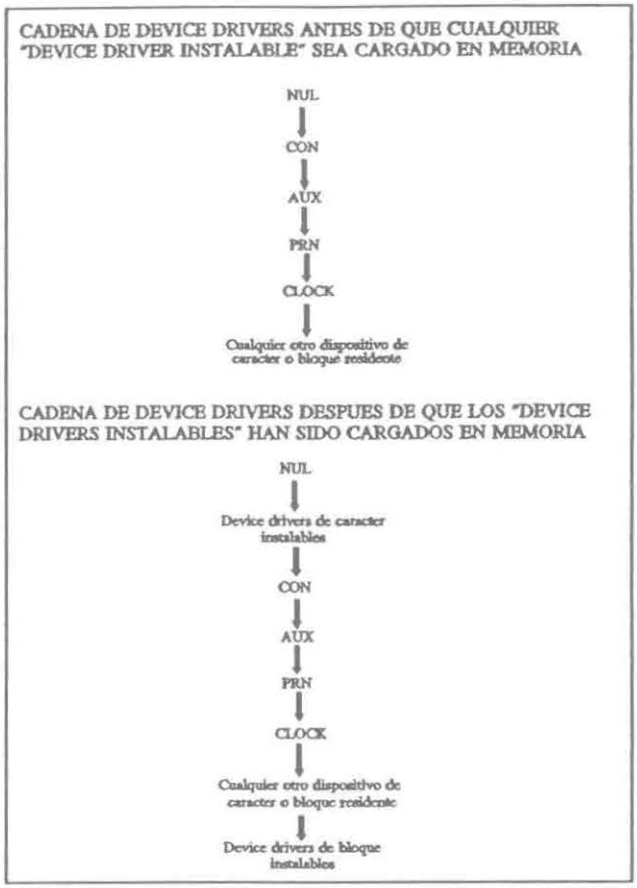


FIGURA IV-2. MANERA COMO EL DOS ENLAZA A LOS DEVICE DRIVERS.

manera similar, los nuevos device drivers con nuevos nombres serán insertados en la lista. Así es como el DOS es capaz de accesar los device drivers.

IV.3. POSIBLES PRESENTACIONES DE LOS DEVICE DRIVERS

Los device drivers de DOS son programas de software que controlan dispositivos, los cuales se vuelven parte del DOS. Debido a que estos programas están escritos siguiendo las especificaciones de diseño de Microsoft, DOS puede reconocer estos nuevos dispositivos e integrarlos con el resto de los dispositivos residentes. Una vez que DOS reconoce las capacidades de cada device driver, los dispositivos pueden ser accesados tan fácilmente como lo es accesar un dispositivo de impresora o de disco.

A continuación se presenta una serie de posibles presentaciones de los device drivers, basada en las funciones que cumplen.

IV.3.1. DEVICE DRIVERS PARA NUEVOS DISPOSITIVOS

Si no existiesen device drivers con una interfase uniforme con el DOS, instalar un nuevo dispositivo a una PC sería una tarea muy difícil. El distribuidor del dispositivo tendría que prover al usuario de una versión modificada del DOS para poder utilizar dicho dispositivo. Esto crearía una serie de problemas que van desde la incompatibilidad a la escasa transportabilidad de las aplicaciones.

Los device drivers son el solución de software más universal

y significativa para controlar dispositivos. Los nuevos dispositivos se convierten en dispositivos estándar en DOS, disponibles para su acceso en cualquier momento, desde dentro de programas de aplicación como fuera de ellos, a nivel del interpretador de comandos (COMMAND.COM).

IV.3.2. DEVICE DRIVERS DE REEMPLAZO PARA DISPOSITIVOS ESTANDAR

Ocasionalmente se puede encontrar el problema de que los device drivers estándar de DOS no cumplen exactamente con lo que el usuario espera de ellos. Para manejar esta situación, el usuario podría reemplazar un device driver existente por uno personalizado. Como se vio con anterioridad el DOS nos permite instalar nuevos device drivers con los nombres viejos de los dispositivos estándar con lo cual el reemplazo es sencillo y directo.

Generalmente, un usuario puede desear agregar nuevos device drivers a DOS por dos razones:

- Para dar soporte a dispositivos del DOS que no son parte del conjunto estándar de dispositivos del DOS (Por ejemplo el popular ANSI.SYS).
- Para reemplazar el device driver original con uno nuevo que pueda tener más capacidad ó transportabilidad que el viejo (Por ejemplo algún device driver para el teclado CON que emule algún teclado para terminal de un mainframe)

IV.3.3. DEVICE DRIVERS SIN DISPOSITIVO

Se ha discutido sobre device drivers para nuevos dispositivos

periféricos y sobre drivers que reemplazan a los del DOS. Existe otro tipo de drivers: aquellos que no controlan a dispositivos de hardware reales. Comúnmente conocidos como **dispositivos virtuales**, estos device drivers simulan un dispositivo de hardware. El ejemplo más claro lo representa un device driver de disco RAM, el cual reserva memoria para simular bytes en el disco y maneja esta memoria como si fuera un disco real. Las lecturas y escrituras a este disco van directamente a memoria en vez de esperar que un disco real accese los datos a una velocidad más lenta.

IV.4. TIPOS DE DEVICE DRIVERS

Los device drivers generalmente son categorizados dentro de dos grupos: device drivers de bloques y device drivers de caracteres. La membresía en uno de estos dos grupos determina como el dispositivo asociado será visto por el DOS y cuales funciones soportará el driver.

IV.4.1. DEVICE DRIVERS DE CARACTERES

Los device drivers de caracteres controlan dispositivos periféricos, como las impresoras (puerto paralelo) ó modems (puerto serial), que realizan E/S un caracter (byte) a la vez. Cada device driver de caracteres generalmente está orientado a atender un solo dispositivo de hardware. El dispositivo tiene un nombre lógico de uno a ocho caracteres que puede ser utilizado por un programa de

aplicación para "abrir" el dispositivo para E/S como si fuera un archivo común y corriente. El nombre lógico se debe estrictamente a motivos de que DOS identifique el driver implicando que el dispositivo no tiene ningún equivalente físico (a diferencia de las etiquetas de volumen que utilizan los dispositivos de bloques)

Los tres device drivers de caracteres residentes para la consola, puerto serial, e impresora (puerto paralelo) llevan los nombres lógicos CON, AUX, y PRN, respectivamente. Estos tres drivers reciben un tratamiento especial del DOS el cual permite que los programas de aplicación direccionen los dispositivos de tres maneras diferentes:

- Pueden ser abiertos con su nombre para E/S (como cualquier otro driver de caracteres).
- Son soportados por llamadas a función de propósito especial del DOS (Interrupción 21h, Funciones 01h - 0Ch).
- Son asignados a handles por default (0 standard input CON, 1 standard output CON, 2 standard error CON, 3 standard auxiliary AUX, 4 standard printer PRN) los cuales pueden ser redireccionados y no necesitan ser abiertos para ser utilizados.

Otros dispositivos de caracter pueden ser soportados con simplemente instalar device drivers de caracteres adicionales. La única restricción sobre el numero total de dispositivos que pueden instalarse, aparte de la memoria necesaria, es que cada dispositivo debe de tener un nombre lógico único.

La manera en como el núcleo del DOS almacene y filtre los

caracteres que pasan entre el y el driver de caracteres dependerá de si el DOS considera que el dispositivo está en modo **cocinado** (COOKED) ó **crudo** (RAW).

Durante la entrada de datos en modo cocinado, el DOS toma los caracteres uno a uno del driver y los coloca en su buffer interno, verificando cada caracter en busca de un Control-C (03h) ó un Return (0Dh). Cuando se ha recibido el número de caracteres pedido por el programa de aplicación ó cuando un Return es detectado, la entrada se termina y los datos son copiados desde el buffer de DOS al buffer del programa de aplicación. Cuando un Control-C se detecta, el DOS aborta la operación de entrada y transfiere el control a la rutina cuya dirección este almacenada en el vector de la Interrupción 23h (Ver capitulo III: INTERRUPCIONES DE SOFTWARE). De manera similar, durante la salida en modo cocinado, el DOS verifica los eventos del sistema entre la transmisión de cada caracter, en busca de un Control-C del teclado para abortar la operación de salida si es detectado alguno.

En el modo crudo, el número exacto de bytes solicitados por el programa de aplicación es leido ó escrito, sin consideraciones hacia ningún caracter de control como el Return ó Control-C. El DOS pasa la petición de E/S al driver en una operación única, en vez de romper la petición en escrituras ó lecturas de un solo caracter, y los caracteres son transferidos directamente de y hacia el buffer del programa solicitante.

El modo para un driver especifico puede ser consultado por un programa de aplicación con la Interrupción 21h, Función 44h,

Subfunción 00h (IOCTL Get Device Data function); además el modo puede ser seleccionado con la Interrupción 21h, Función 44, Subfunción 01 (IOCTL Set Device Data function). En realidad el device driver no está al tanto de su modo y definitivamente el modo no afecta su operación.

En el caso especifico del device driver de caracteres VOX las operaciones se realizan en modo cocinado de salida.

IV.4.2. DEVICE DRIVERS DE BLOQUES

Los device drivers de bloques controlan los dispositivos periféricos que transfieren datos en trozos en vez de hacerlo un byte en cada ocasión. Los dispositivos de bloques son usualmente dispositivos de acceso aleatorio como un drive floppy ó de disco duro, pero también pueden ser dispositivos secuenciales como los drives de cinta magnética. Un driver de bloques puede soportar más de una unidad física y puede mapear dos ó más unidades lógicas dentro de una sola unidad física, como lo hace con un disco duro particionado.

El DOS asigna identificadores de una sola letra (A,B, y los que sigan) a los dispositivos de bloque, en vez de nombres lógicos. La primera letra asignada a un device driver de bloques es determinada solamente por la posición del driver en la cadena de todos los drivers, es decir, por el número de unidades soportadas por el driver de bloques que está antes de el; el número total de letras asignadas a un driver es determinado por el número de unidades lógicas que el driver soporta.

DOS no asocia un modo (cocinado o crudo) a los device drivers de bloques. Un device driver de bloques siempre lee ó escribe el número exacto de sectores solicitados y nunca filtra ó manipula el contenido de los bloques transferidos.

IV.5. ESTRUCTURA DE LOS DEVICE DRIVERS

Como se puede apreciar en la figura IV-3, un programa device driver consiste de cinco partes: Un Device Header, un área de almacenamiento de datos y procedimientos locales, un procedimiento de estrategia, un procedimiento de interrupción, y finalmente las rutinas de procesamiento de comandos. Discutiremos cada una de estas partes en esta sección, pero antes revisaremos que es lo que realiza cada una de ellas.

El principio de un programa device driver no contiene código de la misma manera en que los programas normales. En cambio, el **Device Header** contiene información acerca del mismo device driver. Esta información es utilizada por el DOS e incluye el nombre del driver y un apuntador a el siguiente driver.

FIGURA IV-3. PARTES BASICAS DE UN DEVICE DRIVER.

La siguiente parte del

driver es utilizada para almacenar variables de **datos** locales y **procedimientos locales** ó rutinas.

La tercera y cuarta parte del device driver contiene los procedimientos de **estrategia** e **interrupción.** Estos dos procedimientos se integran para procesar cada comando que es pasado por el DOS al device driver. Ellos permiten que el DOS pase el control completamente al driver.

La quinta y última parte del driver contiene las rutinas que procesarán cada uno de los comandos que DOS le pase al device driver.

IV.5.1. DEVICE HEADER

El device header, ver figura IV-4, siempre se encuentra al principio del driver. Contiene una liga (2 words) hacia el siguiente driver de la cadena de drivers, un word (2 bytes) de banderas de atributos del dispositivo, desplazamientos a las rutinas de estrategia e interrupción, y el nombre lógico del dispositivo si es un dispositivo de caracteres ó el número de unidades lógicas si es un dispositivo de bloques.

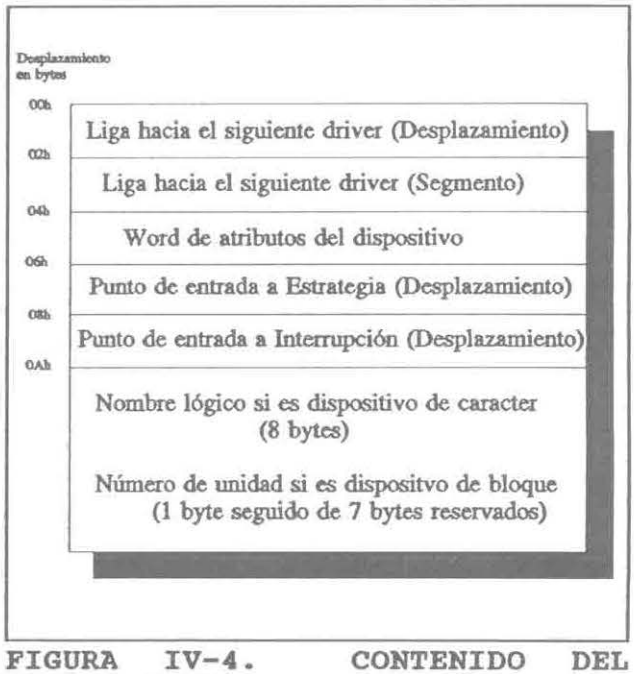


FIGURA IV-4. CONTENIDO DEL DEVICE HEADER.

La word de atributos del dispositivo, ver tabla IV-3, define

BITS FEDC BA98 7654 3210	SIGNIFICADO
....1	Dispositivo de entrada estándar.
....0	Dispositivo de entrada no estándar.
....1.	Dispositivos de caracter: Dispositivo de salida estándar. Dispositivos de bloque: Puede manejar números de sector de 32 bits. (V4)
....0.	Dispositivos de caracter: Dispositivo de salida no estándar. Dispositivos de bloque: No puede (V4) manejar números de sector de 32 bits.
.... .1..	Dispositivo nulo.
.... .0..	Dispositivo no nulo.
.... 1...	Dispositivo de reloj.
.... 0...	Dispositivo no de reloj.
....1	Servicios de driver Int 29h.
.... .000 000.	Reservados antes de V3.2 (poner ceros).
....0.	Reservado en V3.2 (poner cero).
.... .1..	Driver soporta IOCTL (V3.2).
.... .0..	Driver no soporta IOCTL (V3.2).
.... .000 0...	Reservado en V3.2 (poner ceros).
.... 0...	Se soportan medios para ABRIR/CERRAR /REMOVIBLES (V3.2).
.... 1...	No se soportan medios para ABRIR/CERRAR/REMOVIBLES (V3.2).
...0	Reservado (poner cero).
..1.	Dispositivos de caracter: Dispositivo soporta operaciones O-Till-Bussy. Dispositivos de bloque: formato de bloque IBM.
..0.	Dispositivos de caracter: Dispositivo no soporta operaciones O-till-bussy. Dispositivos de bloque: formato de bloque no-IBM.
.1..	Se soporta IOCTL.
.0..	No se soporta IOCTL.
1...	Dispositivo de caracter.
0...	Dispositivo de bloque.
1000 0000 0001 0011	Ejemplo: Para el dispositivo estándar CON: [8013h].
1010 1000 0000 0000	Ejemplo: Para el driver VOX [A800h].
0000 1000 0010 0000	Ejemplo: Para un driver de disco duro IBM con V3.2 [0840h].

TABLA IV-3. CONTENIDO DEL WORD DE ATRIBUTOS.

si un driver controla un dispositivo de caracteres o de bloques, cuales de las subfunciones son soportadas por el driver (dependiendo de la versión de DOS), y, en el caso de los drivers de bloques, si el driver soporta discos duros compatibles con IBM. Los cuatro bits menos significativos del word, controlan si el DOS debe usar el driver como standard input, standard output, reloj, ó dispositivo nulo; cada uno de estos cuatro bits deben de estar encendidos solamente en un driver a la vez en el sistema.

La información en el device header normalmente es usada solamente por el núcleo del DOS y no se encuentra disponible para los programas de aplicación. Sin embargo, las subfunciones del IOCTL (Interrupción 21h Función 44h Subfunciones 00h y 01h) pueden ser utilizadas para inspeccionar ó modificar algunos de los bits en la word de atributos del dispositivo.

IV.5.2. ALMACENAMIENTO DE DATOS Y PROCEDIMIENTOS LOCALES

Aquí se coloca cualquier procedimiento y datos que el device driver pudiera necesitar para completar sus tareas.

En el caso de estar construyendo un device driver de caracteres, esta parte debe de ser tratada con mucha atención y precaución pues es el único lugar donde se podrá proteger las variables de control y de almacenamiento internas del driver, ya que en la transmisión de datos del DOS (hacia o desde el driver), el DOS llama al driver con el mismo comando un número de veces igual a la longitud de la transmisión y no le avisará al driver cuando a terminado de trabajar con el, simplemente dejará de

utilizarlo. Por ejemplo si el DOS va a escribir cien bytes en un device driver de caracteres, el DOS llamará a el procedimiento de interrupción cien veces con el mismo código de procesamiento (escritura), con el fin de completar la transmisión.

En el caso concreto del driver VOX esta área es vital, pues en ella se almacenan localmente todos los caracteres recibidos por intermedio del DOS. Lo cual posibilita la manipulación y análisis de estos datos en el momento apropiado, es decir antes de que DOS deje de llamar al driver.

IV.5.3. PROCEDIMIENTO DE ESTRATEGIA

El DOS llama al procedimiento de estrategia como un primer paso en cualquier operación, pasándole en los registros ES:BX un segmento y un desplazamiento de una estructura de datos llamada el Device Header. El procedimiento de estrategia salva este apuntador para procesamiento subsecuente (por parte de el procedimiento de interrupción), y regresa al DOS.

El Request Header es en esencia un pequeño buffer usado para la comunicación privada entre el DOS y el device driver. Ambos el DOS y el device driver leen y escriben información en el Request Header.

Los primeros trece bytes de un Request Header son los mismos para todas las funciones de un device driver y son por lo tanto referenciadas como la parte estática del header. El número y contenido de los bytes subsecuentes varía de acuerdo al tipo de operación que ha sido requerida por el núcleo del DOS, ver tabla

IV-4. El componente más importante del request header es el código de comando pasado en el tercer byte; este código selecciona una función del driver como la lectura ó escritura.

DESPLAZAMIENTO EN BYTES	LARGO DEL CAMPO	SIGNIFICADO
00h	Byte	Largo del request header.
01h	Byte	Código de unidad: El número del dispositivo, para dispositivos de bloque.
02h	Byte	Código del comando: El número del más reciente comando enviado al driver.
03h	Word	Estatus: Código de estatus puesto por el driver después de cada llamada. Si el bit 15 está prendido, un código de error se encuentra en el byte bajo. Un código de 0 implica una operación exitosa.
05h	8 Bytes	Reservado para el uso del D.O.S.
0Dh	Variable	Datos requeridos por el driver.

TABLA IV-4. CONTENIDO DEL REQUEST HEADER

IV.5.4. PROCEDIMIENTO DE INTERRUPCION

La parte más compleja de un device driver es el procedimiento de interrupción, el cual es llamado inmediatamente después de haber llamado a el procedimiento de estrategia.

Cuando el procedimiento de interrupción recibe el control de DOS, salva todos los registros, examina el request header (cuyo) apuntador fue salvado previamente por el procedimiento de

estrategia), determina cual es el código de comando requerido, y salta a la función apropiada. Cuando la operación se completa, el procedimiento de interrupción guarda el estatus, error (ver tabla IV-5), y cualquier otra información aplicable dentro del request header, restaura el contenido previo de los registro afectados, y regresa al núcleo del DOS.

BITS FEDC BA98 7654 3210	SIGNIFICADO
1... .... 0000 0001	Error: Violación a protección de escritura.
1... .... 0000 0010	Error: Unidad desconocida.
1... .... 0000 0011	Error: Drive no está listo.
1... .... 0000 0100	Error: De CRC.
1... .... 0000 0101	Error: Largo de la estructura request es incorrecta.
1... .... 0000 0110	Error: De búsqueda.
1... .... 0000 0111	Error: Medio desconocido.
1... .... 0000 1000	Error: Sector no encontrado.
1... .... 0000 1001	Error: Impresora sin papel.
1... .... 0000 1010	Error: Falla en escritura.
1... .... 0000 1011	Error: Falla en lectura.
1... .... 0000 1100	Error: Falla general.
1... .... 0000 1101	Reservado.
1... .... 0000 1110	Reservado.
1... .... 0000 1111	Error: Cambio inválido de disco.
....1	Terminación exitosa.
.... ..1.	Dispositivo ocupado.
.xxx xx..	Reservado.
0...	No hay error.

TABLA IV-5. POSIBLE CONTENIDO DEL WORD DE ESTATUS.

La razón de este procedimiento de doble llamada al driver es imprescindible si se desea manejar el concepto de multitarea, ya que en un futuro el DOS podría entre la primera y segunda llamada asignar prioridades a las peticiones de E/S lo cual permitiría realizar más trabajo por período de tiempo.

IV.5.5. PROCESAMIENTO DE COMANDOS

Un total de veinte códigos de comando están definidos para los device drivers del DOS.

CODIGO DEL COMANDO	VERSION DEL D.O.S.	TIPO DE DISPOSITIVO	DESCRIPCION
00h	2+	Ambos	Inicialización.
01h	2+	Bloque	Verificación del medio.
02h	2+	Bloque	Construir BPB.
03h	2+	Ambos	Lectura del control de E/S.
04h	2+	Ambos	Lectura.
05h	2+	Caracter	Lectura no-destructiva.
06h	2+	Caracter	Estatus de entrada.
07h	2+	Caracter	Vaciar buffers de entrada.
08h	2+	Ambos	Escritura.
09h	2+	Ambos	Escritura con verificación.
0Ah	2+	Caracter	Estatus de salida.
0Bh	2+	Caracter	Vaciar buffers de salida.
0Ch	2+	Ambos	Escritura del control de E/S.
0Dh	3+	Ambos	Abrir dispositivo.
0Eh	3+	Ambos	Cerrar dispositivo.
0Fh	3+	Bloque	Medio removible.
10h	3+	Caracter	Salida hasta que este ocupado el dispositivo.
11h-12h	3.2+	-----	No definidos.
13h	4+	Bloque	IOCTL (Control E/S).
14h-16h	3.2+	-----	No definidos.
17h	3.2+	Bloque	Toma dispositivo lógico.
18h	3.2+	Bloque	Establece dispositivo lógico.

TABLA IV-6. CODIGOS DE COMANDO EXISTENTES PARA LOS DEVICE DRIVERS.

Los códigos del comando, las versiones del DOS cuando se

liberaron, así como su descripción se pueden apreciar en la tabla IV-6. Algunas de las funciones son relevantes solamente para dispositivos de caracteres, algunas solo para bloques, y algunas para ambos. De todas maneras, debe de existir una rutina ejecutable presente para cada función, aún y cuando la rutina no haga nada más que prender la bandera de terminación exitosa en el word de estatus del request header.

Las primeras doce funciones deben de ser soportados por el procedimiento de interrupción bajo todas las versiones del DOS. El DOS inspecciona los bits en el word de atributos del dispositivo en el device header para determinar cual de las funciones opcionales soporta el driver. Los requerimientos generales para cada función son descritos con más detalle a continuación.

IV.5.5.1. Inicialización. **[Caracter y bloque]**

El comando 00h es el comando Initialization. DOS siempre llama al device driver con este comando inmediatamente después de que el driver a sido cargado en memoria. Esto permite al device driver realizar sus funciones de inicialización, como escribir un mensaje en el monitor, limpiar registros u otras funciones de instalación. Es importante indicar que esta será la **única** ocasión en la que el driver podrá efectuar llamadas a servicios del DOS, esto es durante su proceso de inicialización. Para todos los demás comandos, el driver **no podrá** generar llamadas a las interrupciones del DOS; si se intenta hacerlo, DOS se perderá en un **agujero negro**, porque el driver es parte del DOS y **el DOS no puede llamarse a si mismo**

(cuando un driver se está inicializando no se considera parte del DOS). Cuando el driver regresa el control al DOS, el DOS asumirá que el driver está listo para procesar otros comandos.

IV.5.5.2. Verificación del medio. **[Bloque]**

El comando 01h es el comando Media Check. Este comando es llamado por el DOS cuando existe pendiente una llamada de acceso a un drive diferente a las simples escritura ó lectura (Por ejemplo, abrir un archivo, cerrarlo, renombrarlo ó borrarlo). Si el comando 01h determina que el disco no ha sido cambiado, DOS procede a accesar el disco. En caso contrario el DOS: invalida todos los buffers asociados con ese drive, realiza una llamada al comando 02h, y lee los datos concernientes al disco.

IV.5.5.3. Construir BPB. **[Bloque]**

El comando 02h es el comando Build BPB. Este comando es llamado por el DOS cuando a recibido un resultado de "cambio en el medio" por parte del comando 01h. La información en el BPB es utilizada por el núcleo del DOS para interpretar la estructura del disco y es también utilizada por el driver para traducir direcciones de sectores lógicos en pistas físicas, sectores, y cabezas.

IV.5.5.4. Lectura del control de E/S. **[Caracter y bloque]**

El comando 03h es el comando IOCTL input. Este comando es utilizado por el device driver para regresar información de control

al programa que llamó al dispositivo. Por ejemplo, si el dispositivo es una impresora, el driver podría regresar información sobre el estatus de la impresora, como la velocidad de transmisión. Aun y cuando esta característica es muy útil, no es una función común en los drivers. Existen dos razones por las cuales no es comúnmente implementado:

- Existe una sola llamada del DOS que permite control de E/S (Interrupción 21h Función 44h Subfunciones 0Ch 0Dh), y muchos programas comerciales no utilizan este servicio, porque no esperan que un driver regrese algún tipo de información.
- Dar esta capacidad a un driver no es tarea sencilla; el driver no sabe que tipo de información regresar.

IV.5.5.5. Lectura. **[Caracter y bloque]**

El comando 04h es el comando Input. Este comando le indica al driver que lea datos del dispositivo. Estos datos son entonces pasados al DOS, el cual los pasa al programa de aplicación que lo llamó.

IV.5.5.6. Lectura no-destructiva. **[Caracter]**

El comando 05h es el comando Nondestructive Input. Este comando es utilizado para determinar ya sea que exista algún dato en el dispositivo, sin pasarlo directamente al programa de aplicación a través del DOS. Esto se realiza a menudo con fines de probar si se está listo para leer del dispositivo. Si existe algún caracter esperando ser leido, simplemente se asigna un comando 04h.

Si no existen caracteres, se le dice al DOS que no hay caracteres listos para ser leidos.

IV.5.5.7. Estatus de entrada. **[Caracter]**

El comando 06h es el comando Input Status. Esta llamada permite al DOS verificar el estatus de un dispositivo. Si el dispositivo no está listo, no se podrá asignar el comando 04h. Por otra parte, si el estatus del dispositivo indica una condición de listo, un comando 04h se puede asignar inmediatamente. Nótese que este comando no es igual al 05h. El comando 06h verifica el estatus del dispositivo; el 06h verifica que exista un caracter en el buffer del dispositivo.

IV.5.5.8. Vaciar buffers de entrada. **[Caracter]**

El comando 07h es el comando Input Flush. Este comando permite descartar cualquier entrada previa del buffer, al limpiar el buffer asociado con el dispositivo. Esto puede ser muy importante para algunos programas de aplicación. Supóngase un programa que pregunte al usuario si desea borrar algunos archivos del disco; si no se asignó previamente una llamada al comando 07h, se podría accidentalmente borrar varios archivos debido a algún caracter (contenido en el buffer) que indique hacerlo. Las llamadas a este comando se deben realizar para asegurarse de que cualquier dato extraño desaparezca del buffer.

IV.5.5.9. Escritura. **[Caracter y bloque]**

El comando 08h es el comando Output. Este comando le indica al device driver que escriba una cantidad específica de datos al dispositivo. De tratarse de drivers bloques le pasa el número de sectores a transferirse. En el caso de drivers de caracteres esta cantidad **siempre es uno**, lo cual obliga a realizar una llamada a el comando 08h por cada caracter que se desee transmitir.

IV.5.5.10. Escritura con verificación. **[Caracter y bloque]**

El comando 09h es el comando Output With Verify. Este comando es similar al anterior pero tiene una función adicional: cuando el switch del comando VERIFY está encendido (ON), el driver leerá los datos después de cada escritura. Por supuesto, aquí se asume que el dispositivo puede leer el mismo dato. Esta particularidad no tiene mucha relevancia para las impresoras ó pantallas, porque estos dispositivos no pueden leer lo que se escribió.

IV.5.5.11. Estatus de salida. **[Caracter]**

El comando 0Ah es el comando Output Status. Este comando le indica al driver que verifique el estatus del dispositivo que se este usando para la salida. Esto no tiene sentido para los dispositivos que no pueden leer datos.

IV.5.5.12. Vaciar buffers de salida. **[Caracter]**

El comando 0Bh es el comando Output Flush. Este comando le indica al driver que envíe una señal al dispositivo, informándole

que cualquier dato que todavía se encuentre en el dispositivo debe ser descartado.

IV.5.5.13. Escritura del control de E/S. [Caracter y bloque]

El comando 0Ch es el comando IOCTL Output. Este comando se envía al driver cuando DOS necesita pasarle datos al driver para uso del mismo driver. Este no es comando que DOS emplea para enviar datos al dispositivo. Si este comando es implementado en el driver, se debe de usar los datos para controlar el dispositivo más que para enviar datos al dispositivo.

IV.5.5.14. Abrir dispositivo. [Caracter y bloque]

El comando 0Dh es el comando Device Open. Este comando puede ser utilizado por el dispositivo para saber cuantas veces a sido abierto. Este comando está disponible en versiones del DOS posteriores a la 3.2. Un driver puede realizar una variedad de funciones cuando es abierto: inicializar variables de control, limpiar áreas internas de memoria, verificar dispositivos, etc.

IV.5.5.15. Cerrar dispositivo. [Caracter y bloque]

El comando 0Eh es el comando Device Close. Este comando es utilizado con el comando 0Dh para implementar una posible contabilización de cuantas veces se ha abierto un dispositivo. Un driver al ser cerrado podría realizar funciones parecidas a las del comando anterior. Este comando está disponible en versiones del DOS posteriores a la 3.2.

IV.5.5.16. Medio removible. **[Bloque]**

El comando 0Fh es el comando Removable Media Command. Disponible unicamente para drivers de bloques, este comando le dice al DOS si el dispositivo contiene medios removibles.

IV.5.5.17. Salida hasta que este ocupado. **[Caracter]**

El comando 10h es el comando Output Till Busy. Este comando es el de más utilidad para las impresoras que tienen un buffer para recibir datos. En vez de transmitir un pequeño número de caracteres, el driver podría enviar suficientes datos para llenar el buffer de la impresora; esto minimizaría el numero de veces que DOS necesita llamar al driver con datos para la impresora.

Los comandos 11h y 12h, no están aún definidos.

IV.5.5.18. IOCTL (Control de E/S). **[Bloque]**

El comando 13h es el comando Generic I/O Control. Este comando es valido para drivers de bloques bajo DOS versión 3.2 ó posterior. Es utilizado por programas de aplicación que utilizan llamadas DOS que permiten servicios de IOCTL (Interrupción 21h Función 44h Subfunción 0Dh). El propósito de este comando es el brindar un servicio estándar para el control de E/S orientado a dispositivos de bloques.

Los comandos 14h,15h y 16h, no están aún definidos.

IV.5.5.19. Toma dispositivo lógico. [Bloque]

El comando 17h es el comando Get Logical Device. Este comando es valido para drivers de bloques bajo DOS versión 3.2 o posterior. Este comando le dice al DOS si existe algún otro letra de drive lógica asignada a la misma unidad física. A partir del DOS versión 3.2, se permite al usuario especificar múltiples letras de drive para una misma unidad de dispositivo (por ejemplo, un disco duro particionado). Es utilizado por programas de aplicación que utilizan llamadas DOS que permiten servicios de IOCTL (Interrupción 21h Función 44h Subfunción 0Eh).

IV.5.5.20. Establece dispositivo lógico. [Bloque]

El comando 18h es el comando Set Logical Device. Este comando es valido para drivers de bloques bajo DOS versión 3.2 o posterior. Este comando permite a los usuarios de DOS especificar múltiples letras de drive para un mismo dispositivo físico. Es utilizado por programas de aplicación que utilizan llamadas DOS que permiten servicios de IOCTL (Interrupción 21h Función 44h Subfunción 0Fh).

IV.6. PROCESAMIENTO DE UNA PETICION TIPICA DE ENTRADA/SALIDA

Cuando un programa de aplicación llama a la Interrupción 21h del DOS para realizar cualquier operación de E/S, los device drivers están implicitos en casi todos los casos (excepto para las llamadas del sistema).

Considérese el ejemplo de una escritura típica a disco

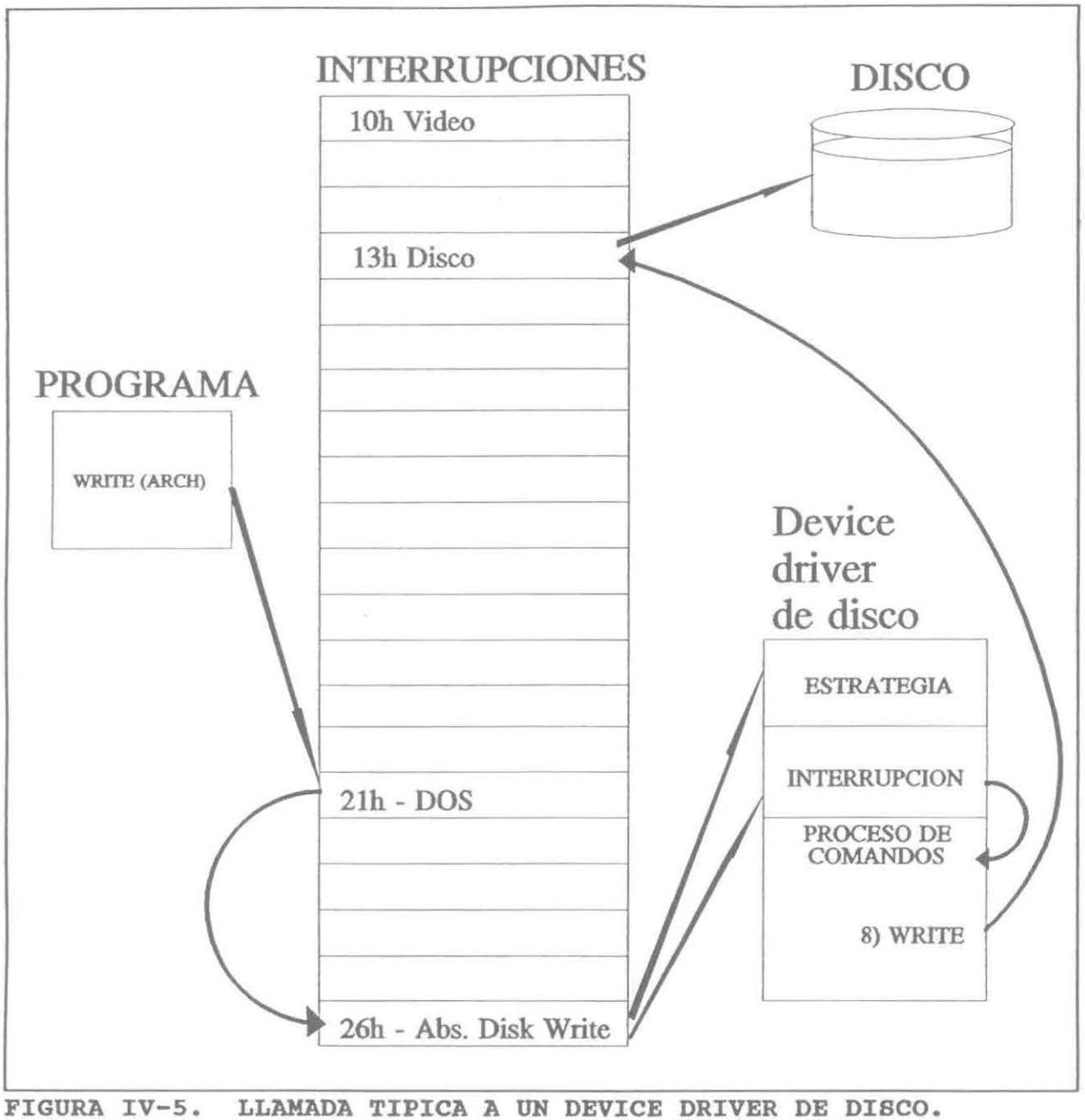


FIGURA IV-5. LLAMADA TIPICA A UN DEVICE DRIVER DE DISCO.

presentado en la figura IV-5, dicha gráfica enseña la secuencia de eventos necesaria para completar la operación. Cada paso implica una transferencia de control a rutinas de más bajo nivel sucesivamente hasta que la escritura en el disco ocurre en realidad.

Todos estos pasos toman lugar para cada operación individual en el disco; nótese que pueden existir varias llamadas al driver para completar cada una de ellas (Los device drivers pueden estar bastante ocupados).

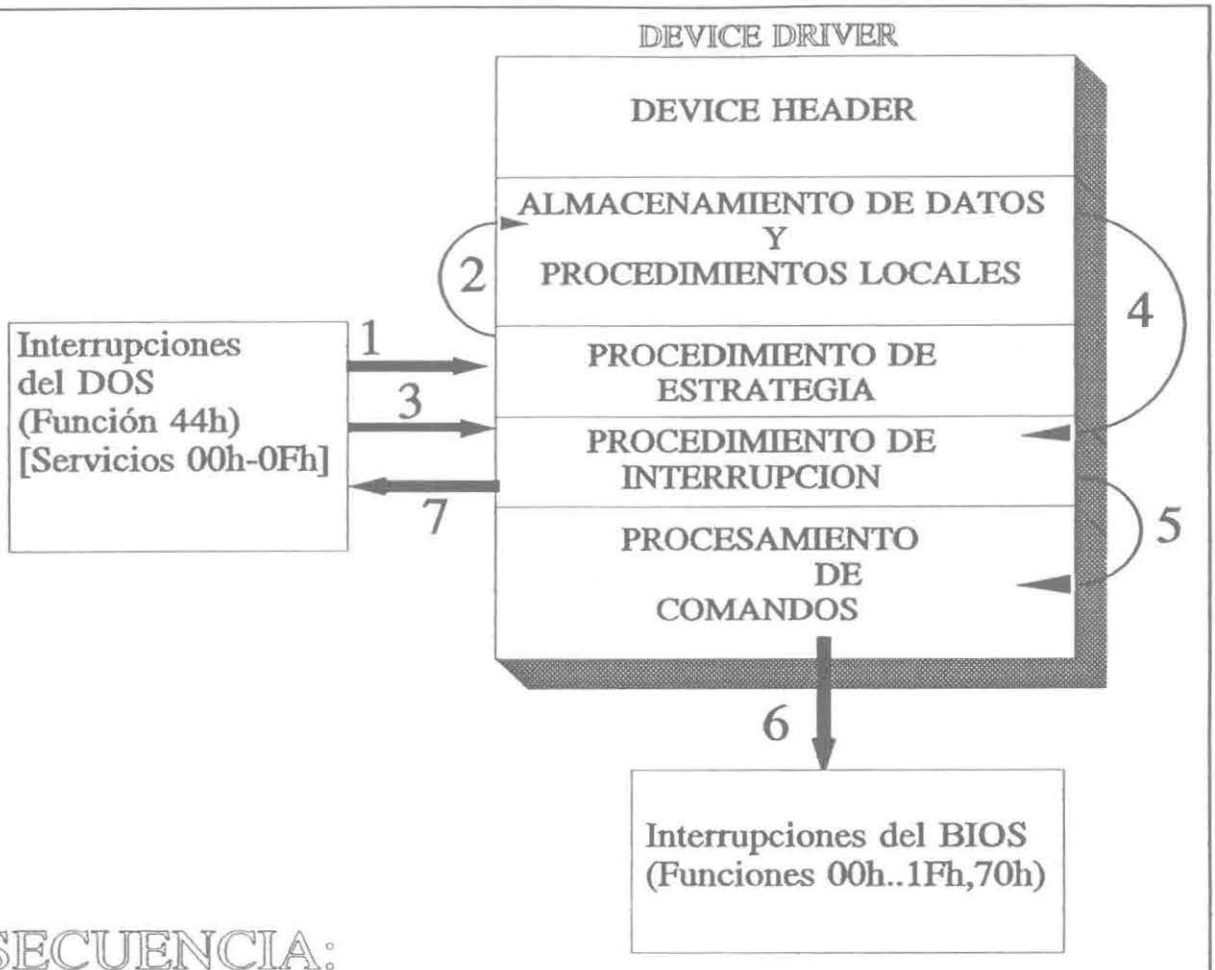
El ejemplo de disco presentado tomó como precedente, el echo de, que el BIOS se hizo cargo de todos los detalles del hardware. Todos los detalles de sincronización, manipulación de bits, y demás son manejados por el BIOS. Para un dispositivo nuevo, el device driver debe de manejar los detalles del hardware directamente la mayoría de las veces.

En la figura IV-6, se puede apreciar lo que normalmente ocurre cuando se accesa a un driver a través de Interrupciones del DOS enfocadas al manejo de device drivers.

IV.7. LOS MECANISMOS PARA LA CONSTRUCCION DE DEVICE DRIVERS

Los device drivers son tradicionalmente codificados en el lenguaje Assembler, debido a los rígidos requerimientos estructurales y por la necesidad de mantener la ejecución de el driver a alta velocidad y con pocos requerimientos de memoria.

Las herramientas mínimas para la construcción de un device driver son similares a las de los programa TSR discutidos en el capítulo anterior (ver tabla III-4). Sin embargo se deben de seguir ciertas consideraciones especiales en la construcción de device



SECUENCIA:

1.- El DOS llama al procedimiento de Estrategia y le entrega ES:BX (Dirección de el Request Header).

2.- Estrategia almacena ES:BX.

3.- El DOS llama al procedimiento de Interrupción.

4.- Interrupción recupera ES:BX y obtiene el código de operación.

5.- Interrupción pasa el control a Proceso de comandos.

6.- Proceso de comandos utiliza el BIOS para trabajar.

7.- Interrupción devuelve ES:BX a DOS, reportando el estatus resultante del comando original.

FIGURA IV-6. FUNCIONAMIENTO DETALLADO DE UN DEVICE DRIVER.

drivers, estas consideraciones se listan en la tabla IV-7 y tienen un carácter imprescindible.

1. Siempre utilice un diskette de prueba para probar los device drivers.
2. Es el procedimiento principal FAR ?
3. En el estatuto ASSUME todos los segmentos apuntan al CODE SEGMENT ?
4. Comienza el device driver en la posición ORG 0h (párrafo) ?
5. Son las estructuras de datos del request header correctas ?
6. El campo de enlace del device header es igual a -1 ?
7. Los bits del campo attribute del device header fueron establecidos correctamente ?
8. Los registros ES y BX son los correctos cuando establece el status word ?
9. Los procedimientos locales salvan todos los registros que utilizan ?
10. Se han reestablecido todos los registros salvados en el stack original ?
11. El device driver tiene el formato ".COM" ?

TABLA IV-7. CONSIDERACIONES IMPORTANTES AL CONSTRUIR UN DEVICE DRIVER.

Después de ensamblar, enlazar y convertir a imagen de memoria, el driver esta listo para ser instalado. Antes de reinicializar el sistema para probar un device driver, es necesario decirle al DOS que cargue el driver en su memoria. Esto se realiza creando (o editando) el archivo llamado CONFIG.SYS el cual reside en el directorio raíz del disco del cual arranca el sistema. La línea que debe incluir este archivo podría ser: DEVICE = VOX.SYS
Lo cual permitirá que al reinicializar el sistema, se carge en la memoria del DOS el device driver VOX.SYS; que en el caso particular del sistema de voz permitirá que el dispositivo pueda ser accesado ya sea como un archivo ó como un dispositivo al que se le envían caracteres tipo ASCII.

Un tópico interesante en el desarrollo de device drivers, es lo referente a la depuración de los mismos. Las herramientas tradicionales para depuración no son de mucha utilidad con los device drivers y la mayoria de las veces los problemas no consisten en errores del programador, sino en algo que el programador no entiende (ó conoce) del driver. El menor de los errores en el procesamiento de los diferentes comandos puede resultar en un bloqueo irrecuperable del sistema. Por lo tanto se recomienda una estrategia de depuración interna del device driver efectuando llamadas al BIOS, ya que el DOS cede todo el control del sistema al device driver cuando lo llama.

CAPITULO V

SUBSTITUCION FONETICA DIRECTA

CAPITULO V

SUBSTITUCION FONETICA DIRECTA

Este capítulo está orientado a facilitar la comprensión de los conceptos: substitución fonética directa, análisis de la cadena hablada, y análisis de inflexión. Para lograr esta comprensión es necesario conocer antes algunas definiciones relacionadas con el lenguaje, fonemas, sonidos, fonología, fonética, y demás conceptos que nos guiarán a través de este capítulo hasta llegar a el propósito original.

V.1. CONCEPTOS PRELIMINARES

En esta sección se presenta una serie de definiciones necesarias para el entendimiento de la substitución fonética directa.

V.1.1. EL LENGUAJE

Se entiende por lenguaje toda **facultad de comunicación.** Cuando el hombre entra en contacto con sus semejantes por medio de signos fónicos, gráficos ó mímicos decimos que se vale del lenguaje oral, gráfico ó mímico. De ello se deduce que el **lenguaje es un**

sistema organizado de signos supositivos.

Es muy importante aclarar la noción de signo. Signos son "cosas" que, por su naturaleza, ó por que se ha convenido así, **pintan en el entendimiento la manera de ser ó el nombre de las cosas.**

El **signo lingüístico** es la unión del significado (una representación) y el significante (palabras de una lengua). Un solo significado admite una multitud de significantes, tantos como idiomas y formas de comunicación puedan existir (por ejemplo: libro, book, livre; tiene un solo significado que es un libro). Estos significantes tienen un doble comportamiento:

- Fónico. Cuando el lenguaje es hablado.
- Escrito. Cuando se utiliza signos gráficos para representar los distintos signos fónicos.

V.1.1.1. Lengua.

Lengua es todo **sistema de signos** propio de una determinada comunidad lingüística. Incluyendo dentro de una comunidad lingüística al conjunto de hombres que hablan la misma lengua. Por lo tanto, hay tantas comunidades lingüísticas como lenguas se habla en el mundo.

V.1.1.2. Habla.

Habla es la **forma personal de expresión** que no cae dentro del sistema de una lengua, porque carece de "sistema", y así como la lengua puede ser resumida en pequeños tratados (gramáticas) que

permiten su enseñanza, el habla no se puede enseñar, ya que hay tantas clases de hablas dentro de una lengua como individuos hacen uso de la lengua.

Así pues, el habla distingue a un hablante, ó a una pequeña comunidad de hablantes, dentro de la grán comunidad que es la lengua.

V.1.1.3. Gramática.

Se llama gramática a la ciencia que **estudia el sistema de una lengua.**

V.1.1.4. Lingüística.

La lingüística estudia todo lo referente al lenguaje, es la **ciencia del lenguaje** en general.

V.1.2. FONEMAS Y SONIDOS

Los fonemas son las **más pequeñas unidades fonológicas** dentro del sistema de la lengua. Así, "cabeza" consta de seis fonemas: /k/-/a/-/b/-/e/-/z/-/a/. Pero hay que tener en cuenta que los fonemas son **unidades ideales**, capaces de ser enseñadas a todos los que empiezan a iniciarse en el aprendizaje de una lengua.

Cuando cada uno de los fonemas es pronunciado por un hablante concreto, y cuando éste pronuncia distintas combinaciones de fonemas que dan como resultado unidades fónicas de orden superior, cada fonema se transforma en un **sonido.**

Recalcando la diferencia: Sonidos son unidades fónicas

producidas por un hablante concreto. Corresponden, como se ve, al plano del habla, mientras que los fonemas caen dentro del plano de la lengua. De ello se deduce que el número de fonemas en una lengua es ilimitado, mientras que hay tantos sonidos como hablantes utilicen una lengua.

Los fonemas se clasifican en:

V.1.2.1. Fonemas vocálicos.

Existe cinco fonemas vocálicos en castellano: /a/, /e/, /i/, /o/, /u/. Estos fonemas vocálicos, ó vocales, han sido producidos con la simple vibración de las cuerdas vocales.

V.1.2.2. Fonemas consonánticos.

Las consonantes se producen no solo con la vibración de las cuerdas vocales, sino con la unión de distintos órganos adscritos al aparato de fonación; así, consonantes como la b y la p precisan la intervención de los labios (bilabiales) que se abren repentinamente al ser emitidas por la voz. Otras, como la t y la d, se producen gracias a la lengua que roza con los dientes del maxilar superior (dentales).

Los fonemas consonánticos del castellano son:

/b/ (igual a la "v", "w"),

/c/ (igual a la "z"),

/ch/, /d/, /f/,

/g/ (ante las letras: a, o, u),

/j/ (que es la "g" ante las letras: e, i),

/k/ (cuando a la letra "c" se le suman: a, o, u),

/k/ (cuando a la "qu" se le suman: e, i),

/l/, /ll/, /m/, /n/, /ñ/, /p/, /r/, /rr/, /s/, /t/, /u/,

/y/ (ya, ye, yi, yo, yu = y, que es una semiconsonante),

/x/ (que es igual a "c+s").

V.1.3. LA FONOLOGIA

La fonología estudia todo lo referente a los fonemas y a su **función dentro de una lengua.** Así pues, estudiará las combinaciones de fonemas para producir unidades fonológicas superiores: sílabas, palabras, etc.

V.1.4. LA FONETICA

La fonética **estudia los sonidos**, es decir, los fonemas actualizados, emitidos por un hablante concreto. Le interesan las cualidades físicas del sonido (timbre, tono, intensidad) y el mecanismo fisiológico que ha permitido su producción (si los sonidos han sido producidos gracias a determinada posición de los labios, los dientes, ó a la vibración de las cuerdas vocales, etc). Hay que tener en cuenta que **no todos los signos fónicos corresponden claramente a un determinado signo gráfico** (como veremos más adelante), es decir el alfabeto fonológico ó el que transcribe fonéticamente los distintos sonidos no corresponde con el llamado alfabeto ortográfico (que es el de los caracteres ASCII que se analizan en el sistema de voz).

V.2. ORTOLOGIA DE LOS FONEMAS

Ortología es la parte de la gramática de una lengua que enseña la **correcta pronunciación de sus fonemas.**

Veremos a continuación como se pronuncian las consonantes que ofrecen ciertas dificultades:

- El signo gráfico "C" equivale a los fonemas /c/, /k/, /qu/.

  c + a, o, u = k: casa = kasa; cosa = kosa; cuna = kuna

  qu + e, i = k: queso = keso; quiero = kiero

- El signo gráfico "C" seguido de e, i, equivale al fonema /z/.

  Cecilia = Ze-zi-lia; Círculo = Zír-cu-lo

- En castellano el signo gráfico "B" y el signo gráfico "V" representan un solo fonema, el /b/.

Se pronuncian igual: revelar (una fotografía) y rebelar (sublevarse).

- En castellano los signos gráficos dobles "CH" y "LL" responden a un fonema sencillo: cacho, coche; llena, llana.

- El signo gráfico "X" equivale a c + s: examen = ecsamen.

- La "G" suena como j delante de los fonemas /e/,/i/: general, geranio, gigante gitano.

Y la "G" suena como g suave delante de los fonemas /a/, /o/, /u:

gata, gota, gula.

Cuando se desea que suene suave ante e, i es preciso escribir "GU": guerra, guitarra , aguijón.

- La "RR" siempre suena fuerte: carro, carrera.

La "R" suena fuerte cuando es inicial ó después de consonante: Rata, enredo.

La "R" es suave cuando es intervocalida: araña, harina, garabato.

La "R" es liquida cuando esta después de (b, c, g, t, g): brazo, ladrón, trozo, cráter, gracias.

V.2.1. CORRESPONDENCIA ENTRE FONEMAS Y LETRAS

Por convención se llama letras a los signos gráficos que sirven para representar los distintos fonemas. El conjunto de estos signos gráficos recibe el nombre de alfabeto gráfico y tiene veintiocho signos ó letras. Por ejemplo:

- Al fonema B le corresponden tres signos gráficos: b, v, w.
- Al fonema K: qu, c (+ a, o, u).
- Al fonema Z: z (de garza) y c (+ e, i).
- Al fonema J: j (de jirafa, jarro, tarjeta) y g (de general, vigilante)
- Al fonema G suave: g (g + a, o, u)

V.3. ¿ QUE ES LA SUBSTITUCION FONETICA DIRECTA?

La substitución fonética directa es el método que se utiliza en el sistema de voz para poder generar sonidos a partir de letras. Formalmente hablando, es el proceso necesario para la selección de un fonema correspondiente a una letra en particular, la selección mencionada obedecerá a el contexto (posición y acompañamiento en la cadena escrita) en el cual se halla la letra.

Existen dos herramientas que complementan a la substitución fonética directa, estos son:

V.3.1. ANALISIS DE LA INFLEXION

Este análisis se enfoca a un caso particular de los fonemas: "los acentos". La existencia de un acento en el sistema de voz implica la acción de elevar la intensidad ó el tono de una sílaba en el seno de la palabra (es decir, dentro de la palabra).

V.3.2. ANALISIS DE LA CADENA HABLADA

El análisis de la cadena hablada se refiere a la observancia de la temporarización implícita al pronunciar una frase. Concretamente se enfoca a la asignación de tiempos, intensidades, acentuaciones, y continuidades provocados por signos de admiración, pregunta, comas, etc.

CONCLUSIONES

Definitivamente el DOS es un sistema operativo que brinda una plataforma muy estable para el desarrollo de nuevas aplicaciones.

El sistema de voz es una aplicación concreta en la que varios factores estuvieron involucrados para el éxito de su implementación, entre esos factores se encuentran: la arquitectura abierta de las PC y la flexibilidad de adecuación del sistema operativo DOS a los nuevos dispositivos.

Es en este último aspecto donde el DOS es poderoso, pues al permitir la creación, instalación y uso de nuevos manejadores de dispositivos, los límites del desarrollo solo están definidos por la imaginación del usuario.

El desarrollo de los device drivers es un tópico muy delicado en el cual el aspecto más importante radica en la comprensión de la filosofía de su funcionamiento, sin la cual no se podría llegar a mayores resultados.

La simulación de multitarea es otra característica notable de este medio ambiente, en donde los riesgos están balanceados con el profesionalismo de los resultados que se pueden obtener.

La substitución fonética directa es un concepto bastante interesante, el cual nos permite realizar la generación formal de fonemas a partir de una cadena de caracteres analizada.

APENDICES:

VOX.SYS
VOXTSR.COM
EA.EXE

APENDICE A

NOMBRE DEL PROGRAMA

VOX.SYS

TIPO

Imagen de memoria; No ejecutable.

CARACTERISTICAS

Es un device driver escrito en Assembler, cuyo word de atributos 0A800h le asigna las siguientes características:

- Dispositivo de caracteres (bit 15 encendido).
- Salida hasta que este ocupado (bit 13 encendido).
- Se soporta Abrir /Cerrar / Medio removible (bit 11 encendido)

HERRAMIENTAS UTILIZADAS PARA SU DESARROLLO

- Microcomputadora 80286, con 1Mb de memoria RAM, disco duro de 42 MB, y monitor EGA color de alta resolución.
- Editor de Turbo Pascal 5.0 de Borland.
- Compilador de Macro Assembler de Microsoft.
- Enlazador de código objeto Link de Microsoft.
- Utilería EXE2BIN.COM de Microsoft.

CONFIGURACION MINIMA REQUERIDA PARA SU FUNCIONAMIENTO

- Computadora 80x86, con 640Kb de memoria, disco floppy, puerto paralelo, y monitor monocromático.

FUNCION QUE DESEMPEÑA

Este programa es el encargado de recibir una secuencia de caracteres ASCII, y procesarlos de acuerdo a dos criterios:
1] Si se trata de un texto que se desea hablar, los caracteres son capturados uno por uno (restricción del DOS), almacenados en un área local de memoria, y al detectarse un fin de archivo (esto se logra al adelantarse a la transmisión actual del DOS) se envían a la rutina de substitución fonética directa ,la cual obtiene y almacena los fonemas correspondientes a la cadena de caracteres capturada. Antes de devolverse el control al DOS, el device driver verifica si está presente el TSRVOX.COM (revisa firma en memoria), de ser así le copia al TSR los fonemas obtenidos y activa explícitamente al TSR; de no encontrarse en memoria el TSR, el propio device efectúa la transmisión al diseño de hardware y se informa al usuario de la falta del TSR (por medio del monitor).
2] Si se trata de la actualización de fonemas y reglas, el programa captura los caracteres hexadecimales correspondientes a los nuevos fonemas y reglas. Simultáneamente a la captura los va actualizando en la tablas (de fonemas y reglas) originales del device. Este proceso está orientado a trabajar en conjunto con el programa EA.EXE unicamente (debido a la estricta estructura de memoria implicada en el manejo de tablas).

FUNCIONAMIENTO

Gracias a su naturaleza de device driver de caracteres, este programa puede se utilizado de diversas maneras, entre las más comunes se encuentran:

- Como redireccionamiento de un archivo ASCII hacia un dispositivo (en este caso VOX). Por medio de instrucciones de DOS en el formato:

  C:>COPY {nombre del archivo} VOX

  C:>TYPE {nombre del archivo} > VOX

- Como petición directa al sistema operativo, cargando los registros adecuados y accesando la interrupción 21h, función 3Dh (abrir un archivo), función 40 (escribir a un archivo), y función 3Eh (cerrar un archivo). Este enfoque se puede utilizar desde lenguajes que permitan realizar llamadas al sistema operativo, por ejemplo: Assembler, C, BASIC, ó Pascal.
- Desde un programa de aplicación elaborado en un lenguaje de alto nivel, efectuando una manipulación de un archivo de texto (compuesto de caracteres ASCII), empleando instrucciones estándar del lenguaje que se este utilizando. Por ejemplo en Turbo-C, se pueden utilizar las instrucciones: _open (abrir un archivo), _write (escribir a un archivo), _close (cerrar un archivo).

APENDICE B

NOMBRE DEL PROGRAMA

TSRVOX.COM

TIPO

Imagen de memoria; ejecutable.

CARACTERISTICAS

Este en programa TSR híbrido escrito en Assembler, el cual tiene una mínima interfase con el usuario (mensajes en línea de error).

HERRAMIENTAS UTILIZADAS PARA SU DESARROLLO

- Microcomputadora 80286, con 1Mb de memoria RAM, disco duro de 42 MB, y monitor EGA color de alta resolución.
- Editor de Turbo Pascal 5.0 de Borland.
- Compilador de Macro Assembler de Microsoft.
- Enlazador de código objeto Link de Microsoft.
- Utilería EXE2BIN.COM de Microsoft.

CONFIGURACION MINIMA REQUERIDA PARA SU FUNCIONAMIENTO

- Computadora 80x86, con 640Kb de memoria, disco floppy, puerto paralelo, y monitor monocromático.

FUNCION QUE DESEMPEÑA

Inicialmente el TSR verifica que no esté presente en memoria, de no encontrarse a si mismo, el programa reserva un área de memoria (que contendrá los fonemas que el device driver VOX le copie), se instala a si mismo en el vector de interrupción 08h (pulso de tiempo) y reprograma el reloj del sistema a una base de tiempo más apropiada para el sistema de voz; interrumpiendo al procesador en cada ocasión que el TSR necesita enviar bytes al diseño de hardware

FUNCIONAMIENTO

El funcionamiento de este programa es totalmente interno, por lo tanto solo basta con llamarlo a ejecución bajo su nombre en la línea de comandos del DOS.

APENDICE C

NOMBRE DEL PROGRAMA

EA.EXE

TIPO

Ejecutable.

CARACTERISTICAS

Este es un programa de aplicación ejecutable escrito en el lenguaje C, el mismo que ofrece una interfase amigable con el usuario.

HERRAMIENTAS UTILIZADAS PARA SU DESARROLLO

- Microcomputadora 80286, con 1Mb de memoria RAM, disco duro de 42 MB, y monitor EGA color de alta resolución.
- Editor de Turbo-C 2.0 de Borland.
- Compilador de Turbo-C 2.0 de Borland.

CONFIGURACION MINIMA REQUERIDA PARA SU FUNCIONAMIENTO

- Computadora 80x86, con 640Kb de memoria, disco floppy, puerto paralelo, y monitor monocromático.

FUNCION QUE DESEMPEÑA

Este programa esta orientado a la **edición y análisis de**

fonemas y reglas. El programa permite la modificación y prueba interactiva de los fonemas y reglas contenidas en el device driver VOX, posibilitando la substitución permanente (hasta que se reinicialize el sistema) de las tablas de fonemas y reglas internas del device driver. Adicionalmente se permite guardar en archivos (de fonemas y reglas del sistema) el contenido de una sesión de trabajo, lo cual brinda una alta flexibilidad cuando se desea cambiar rápidamente el comportamiento del sistema de voz en ocasiones posteriores.

FUNCIONAMIENTO

El programa presenta cinco modos de ejecución en la línea de comandos del DOS:

- C:>EA {una sola palabra}

Si se desea escuchar rápidamente una palabra por la bocina (No es necesario entrar al programa).

- C:>EA "{una frase de 132 caracteres}

Si se desea escuchar una frase completa por la bocina (No es necesario entrar al programa).

- C:>EA F {nombre de archivo de fonemas}

Si se desea actualizar los fonemas del device driver VOX, con el contenido de un archivo de fonemas del usuario (No es necesario entrar al programa).

- C:>EA R {nombre de archivo de reglas}

Si se desea actualizar las reglas del device driver VOX, con el contenido de un archivo de reglas del usuario (No es necesario entrar al programa).

- C:>EA

Si se desea entrar al medio ambiente de ventanas que ofrece el programa editor analizador de fonemas y reglas.

BIBLIOGRAFIA

DOS Programmer's reference

[Terry Dettmann] 1989 {QUE CORPORATION}

Advanced MS-DOS Programming

[Ray Duncan] 1989 {MICROSOFT PRESS}

The MS-DOS Encyclopedia

[Ray Duncan] 1988 {MICROSOFT PRESS}

Writing MS-DOS device drivers

[Robert S. Lai] 1989 {ADDISON WESLEY}

Microsotf Macro Assembler (Programmer's Guide Version 5.1)

[Microsoft] {MICROSOFT PRESS}

Using Assembly Languaje

[Allen L. Wyatt] 1990 {QUE CORPORATION}

Turbo C++ Bible

[Naba Barkakati] 1990 {SAMS}

Manual de ortografía moderna

[Angeles Cardona] 1980 {BRUGUERA}

GLOSARIO

80X86.- Familia de procesadores fabricados por la compañía INTEL.

ASCII.- (American Standard Code for Information Interchange) Código americano estándar para el intercambio de la información.

ASSEMBLER.- Lenguaje pseudo-inglés, llamado código fuente, que es escrito en forma de instrucciones cortas fácilmente traducibles a lenguaje maquinal.

BANDERAS.- Es un indicador usado para señalar otras funciones del hardware ó del software.

BIOS.- (Basic Input Output System) Sistema básico de entrada / salida.

BIT.- Unidad mínima de almacenamiento de datos, el cual puede presentar dos estados (cero y uno).

BPB.- (BIOS Parameter Block) Bloque de parámetros del BIOS.

BUFFER.- Lugar en memoria RAM destinado al almacenamiento de datos temporal.

BUS.- Dispositivo físico utilizado para la transmisión de datos.

BYTE.- Unidad básica de almacenamiento y manipulación. Un byte es equivalente a 8 bits y puede contener un valor en el rango de 0 a 255.

CHIPS.- Oblea de silicio que contiene circuitos integrados en gran escala.

COMMAND.COM.- Shell estándar del DOS.

CONFIG.SYS.- Archivo de configuración del sistema operativo DOS.

CONTROL-C.- Combinación de teclas que ordena al procesador la terminación de una tarea.

CS.- (Code Segment) El registro del código.

DEVICE DRIVER.- (Conductor de dispositivo) Clase particular de programa residente en memoria que pasa a formar parte del sistema operativo DOS, este programa permite la utilización de dispositivos físicos en una PC.

DOS.- (Disk Operative System) Popular sistema operativo de disco distribuido por Microsoft e IBM.

DRIVE.- Dispositivo físico de disco.

FAR.- Procedimiento que salva la dirección de retorno completa (el contenido del CS:IP) en el stack. Se asume que se encuentra en un segmento de código diferente al actual.

HANDLE.- (Manejador) Es un apuntador activo a un dispositivo.

HARDWARE.- Se refiere a las partes físicas (electrónicas y magnéticas) de una computadora.

IBM.- (International Bussines Machines) Compañía líder en el mercadeo de Hardware.

INTEL.- Compañía líder en la fabricación de procesadores de la familia 80x86.

IO.SYS.- Archivo del MS-DOS que contiene las rutinas del BIOS.

IOCTL.- (Input Output Control) Control de entrada / salida orientado a los dispositivos de una PC.

IP.- (Instruction Pointer) El registro apuntador a la instrucción

actual del sistema.

IRQ.- (Interrupt ReQuest level) Nivel de la petición de interrupción.

ISR.- (Interrupt Service Routine) Rutina de servicio a una interrupción.

MICROSOFT.- Compañía líder en el mercadeo de software.

MSDOS.SYS.- Archivo del MS-DOS que contiene el núcleo del sistema.

NUL.- Dispositivo nulo del DOS utilizado para comenzar la cadena de dispositivos disponibles en el sistema.

PC.- (Personal Computer) Computadora Personal.

PROCESADOR.- (Sinónimo de microprocesador) Chip que representa la unidad central de proceso (CPU) de una computadora.

PSW.- (Program Status Word) Conjunto de bits que definen el estatus de un programa particular.

REGISTROS.- Las áreas de almacenamiento de datos, tienen 16 bits de largo y son utilizadas por el procesador para realizar operaciones.

RETURN.- Caracter que representa el retorno de carro.

ROM.- (Read Only Memory) Memoria solamente de lectura, conocida también como Firmware.

RAM.- (Random Access Memory) Memoria de acceso aleatorio.

SHELL.- Programa procesador de comandos en línea, el cual está orientado a la interfase con el usuario de un sistema computacional.

SOFTWARE.- Se refiere a los programas (de aplicación y operativos) que se utilizan en las computadoras.

STACK.- Un área de memoria separada para el almacenamiento

temporal de los valores del ambiente computacional. Esta estructura opera con una filosofía de "el último que entra es el primero que sale".

STDAUX.- (Standard Auxiliary) Dispositivo auxiliar estándar de una PC.

STDIN.- (Standard Input) Dispositivo de entrada estándar de una PC, generalmente es el teclado.

STDOUT.- (Standard Output) Dispositivo de salida estándar de una PC, generalmente es el monitor.

STDPRN.- (Standard Printer) Dispositivo de impresión estándar de una PC, generalmente es la impresora.

TSR.- (Terminate and Stay Resident) Programas que se terminan de ejecutar y quedan residentes en memoria.

WORD.- Representa, dos bytes (16 bits) consecutivos de datos.