Escuela Superior Politécnica del Litoral

Facultad de Ingeniería Eléctrica

Diseño y Construcción de un Equipo de Telex Basado en un Computador Personal IBM PC/XT/AT/PS2

Tesis de Grado

PREVIA A LA OBTENCION DEL TITULO DE:

Ingeniero en Electricidad

Especialización. ELECTRONICA

Presentada por:

Freddy Enrique Pinto Carpio

Guayaquil - Ecuador

1991

A G R A D E C I M I E N T O

Mi especial agradecimiento al Ing. JAIME PUENTE PEREZ.,
Director de tesis.
Por su valiosa ayuda académica el el desarrollo de este proyecto.

A los compañeros de Facultad, y a todas las demás personas que hayan contribuido en la realización de este trabajo.

D E D I C A T O R I A

A MI MADRE,

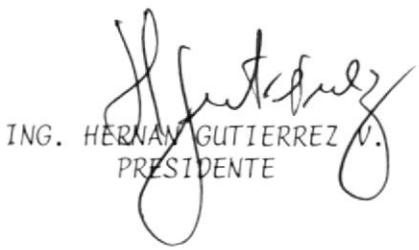
Pilar de mi formación, quién con su exfuerzo y sacrificio supo guiarme por el buen camino del estudio y de cuyas enseñanzas viviré eternamente agradecido.

A MI ESPOSA,

Un justo reconocimiento a ella por el empuje y apoyo brindado en la consecuencia de mi objetivo.

A MIS HIJOS: MIRIAM Y FREDDY

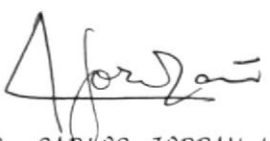
A MIS HERMANOS



ING. HERNAN GUTIERREZ V.
PRESIDENTE

ING. JAIME PUENTE P.
DIRECTOR TESIS

ING. HUGO VILLAVICENCIO V.
MIEMBRO PRINCIPAL



ING. CARLOS JORDAN V.
MIEMBRO PRINCIPAL

D E C L A R A C I O N E X P R E S A

" La responsabilidad por los hechos, ideas y doctrinas expuestas en la presente tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Titulos profesionales de la ESPOL).

FREDDY PINTO CARPIO.

INDICE

2. DESCRIPCION GENERAL DE LA FAMILIA IBM PC

2.1 Introducción.

2.2 El Microprocesador 8088.

2.3 Los Controladores de Periféricos.

2.4 Cómo se comunica el 8088.

2.5 Direccionamiento de memoria en el 8088.

2.6 Juego de Registros del 8088.

2.7 Uso de los Puertos de Entrada/Salida.

2.8 Interrupciones del 8088.

3. PROGRAMACION EN LENGUAJE ENSAMBLADOR DEL MICROPROCESADOR 8088.

3.1 Introducción.

3.2 Formato de las declaraciones en Lenguaje Ensamblador.

3.3 Modelo para un programa en Lenguaje Ensamblador.

3.4 Instrucciones de Movimientos de Datos.

3.5 Modos de direccionamiento.

3.6 Saltos y llamados a Subroutinas.

3.7 Interrupciones de Programas.

3.8 Saltos Condicionales.

3.9 Instrucciones Aritméticas.

3.10 Instrucciones Lógicas.

3.11 Instrucciones de Cadenas.

PREFACIO

La presente tesis tiene como objetivo principal convertir el computador personal IBM PC en un equipo teleimpresor para el envío y recepción de mensajes a través de la línea de TELEX suministrada por IETEL.

El sistema consta de lo siguiente:

- Un computador de la familia IBM PC/XT/AT/PS y/o compatibles.

- Una impresora

- Una interfase que permita interpretar los mensajes de recepción desde la línea de TELEX.

- Un puerto de comunicación serial.

- El programa necesario para controlar el envío y recepción de los mensajes de TELEX.

De acuerdo a las especificaciones dadas, cualquier computador perteneciente a la familia IBM PC (incluida la nueva familia de los PS/2 y compatibles) puede ser utilizado como un equipo de teleimpresión de mensajes; suficiente con que tenga un puerto de comunicación serial instalado como COM1 o COM2, el

mismo que es interconectado con una interfase especialmente diseñada para comunicarse con la línea de TELEX.

Bajo cualquier condición de trabajo que se esté realizando en el computador, el sistema está en la capacidad de recibir mensajes de télex, es decir, por medio de la ejecución de una interrupción de la circuitería del computador, cualquier aplicación que se esté ejecutando, puede ser interrumpida para aceptar el envío de mensajes de télex. Una vez que el mensaje haya sido enviado en su totalidad, el sistema podrá retornar a sus condiciones iniciales de trabajo sin que sufra modificaciones algunas. Por otro lado, si un mensaje está por ingresar al computador, éste lo recibirá sin necesidad de que la aplicación que se esté ejecutando sea suspendida.

Todo mensaje de télex enviado o recibido puede ser pasado a la impresora, indicando la fecha, hora y tiempo de conexión con la línea de télex. Además todo télex recibido puede almacenarse en el disco como un archivo con extensión TLX.

Existe la opción del modo conversacional entre diferentes usuarios; es decir, el sistema permite que dos abonados de télex puedan comunicarse y realizar un diálogo, intercambiando frases.

Se ha elaborado un programa que permite ejecutar cada una de las opciones señaladas. TELEX.COM es un programa que puede ser ejecutado una sola vez desde la línea de comandos del DOS. Una vez cargado en memoria, TELEX inicializa el puerto

de comunicación disponible para uso con la línea de télex (el cual puede ser COM1 o COM2) y reserva cierto espacio de memoria que es utilizado por un procesador de palabras interno. Finalizado este proceso, TELEX devuelve el control del sistema al DOS, pero permanece en modo residente en memoria y espera pacientemente por la combinación ALT-SHIFT-IZQ para ser activado y realizar todas las funciones que ofrece.

Si durante la ejecución de una aplicación, el usuario presiona la secuencia ALT-SHIFT-IZQ, TELEX se activa automáticamente, mostrando una ventana con todas sus opciones disponibles. En este momento, el usuario puede:

- Crear un texto para el envío por la línea de comunicación.
- Modificar un texto que haya sido previamente elaborado.
- Imprimir un texto de télex.
- Enviar un texto por la línea de comunicación
- Recibir mensajes de télex, los mismos que son grabados como archivos en cierto directorio de disco especificado en la instalación del programa.
- Realizar diálogos entre diferentes abonados de télex
- Ejecutar algunos de los comandos internos del DOS, tales como: Visualizar el contenido de los mensajes que han sido grabados como archivos de disco, sean estos mensajes enviados o mensajes recibidos; borrar archivos,

renombralos, moverlos desde un directorio a otro, editarlos para su modificación, asi como visualizar el listado de un directorio de disco y el cambio de directorios y unidades de disco.

La elaboración de la presente tesis ha sido dividida en cinco capítulos, los cuales detallan el procedimiento seguido para el cumplimiento de los objetivos propuestos. Inicialmente se indica una breve introducción de todo el sistema a implementarse, los objetivos de la tesis y el equipo a utilizarse.

El capítulo 1 proporciona un detalle de cómo se realiza la transmisión de información a través de la línea de télex explicando el modo de transmisión asincrónica, el código utilizado en la transmisión de mensajes de télex y un detalle de cómo se realiza una comunicación usando un computador personal.

En el capítulo 2 se detalla la arquitectura del computador PC/XT así como el microprocesador utilizado por el sistema, es decir, el microprocesador 8088.

El capítulo 3 suministra la información necesaria para la elaboración del programa en Lenguaje Ensamblador utilizado para el envío y recepción de mensajes.

El capítulo 4 se refiere al diseño de la interfase que comunica al computador con IETEL, la circuitería para la conversión paralelo a serial de salida de datos y de serial a

paralelo de recepción de datos. Aqui se efectua una descripción del uso del UART 8250 y su forma de programarlo.

El último capítulo comprende todo el programa necesario para el uso del sistema, utilizando el Lenguaje Ensamblador explicado en el capítulo 3.

Por último se proporciona una serie de conclusiones y recomendaciones así como mejoras que puedan ser aplicadas.

CAPITULO UNO

FUNDAMENTOS DE LA COMUNICACION DE DATOS

1.1 INTRODUCCION

La introducción del Computador Personal IBM generó un gran interés en el mundo de las computadoras. Una de las utilidades del PC es la comunicación de datos. Además de leer y escribir datos a dispositivos tales como la pantalla de video o dispositivos de almacenamiento, los computadores a veces necesitan comunicarse con impresoras, terminales u otros computadores, ya sea en forma directa o a través de una línea telefónica.

Este tipo de entrada/salida es típicamente denominado COMUNICACION DE DATOS, puesto que el flujo de información digital es transferido desde y hacia el computador. Este flujo de información debe estar sincronizado para evitar la pérdida de datos lo cual se realiza mediante líneas de control o por medio de secuencias de caracteres especiales.

Un ejemplo sencillo puede ser la comunicación entre dos computadores los cuales tienen diversas responsabilidades, tales como servir al teclado, pantalla o unidades de disco e impresoras. Los datos pueden transferirse entre ambos

computadores a la más alta velocidad de transmisión que pueda proporcionar la circuitería de comunicación, pero en ciertos momentos el computador de recepción puede estar muy ocupado escribiendo al disco o a la pantalla, o leyendo desde el teclado. para lo cual necesita indicar al computador de transmisión que detenga el envío hasta que termine de atender sus servicios locales. .

En el presente capítulo se explican los términos empleados en la comunicación de datos. Es importante reconocer los términos y las siglas que aquí se citan, ya que en la actualidad son muy habituales en la industria. Bajo el tema "Comunicación de datos", se introducen definiciones técnicas relativas a la comunicación.

1.2 LA TRANSMISION DE INFORMACION

La transferencia de información entre dos sistemas digitales como por ejemplo un computador y un terminal periférico u otro computador, se realiza generalmente caracter a caracter utilizando códigos binarios (ASCII, EBCDIC, BAUDOT...). Otras veces, la información que se transfiere no corresponde a ninguna codificación de caracteres numéricos o alfanuméricos sino que es puramente binaria, por ejemplo, cuando se efectúan cargas de programas objeto sobre la memoria del computador.

De una forma o de otra, la información se transmite en unidades de información denominadas palabras, que suelen ser

de 5 a 8 bits. Existen dos formas de realizar la transmisión de estas palabras: el método paralelo y el método serial.

1.2.1 Método Paralelo

Transmitiendo simultáneamente por líneas separadas, todos los bits de la palabra, junto con una señal de reloj que indica el momento en que está presente una palabra de información en las líneas de datos. (Fig 1.1).

1.2.2 Método Serial

Transmitiendo en forma secuencial en el tiempo todos los bits de la palabra, uno tras otro, por una sola línea de datos. Eventualmente puede existir una línea de reloj que marca todos los tiempos de bit (Fig 1.2).

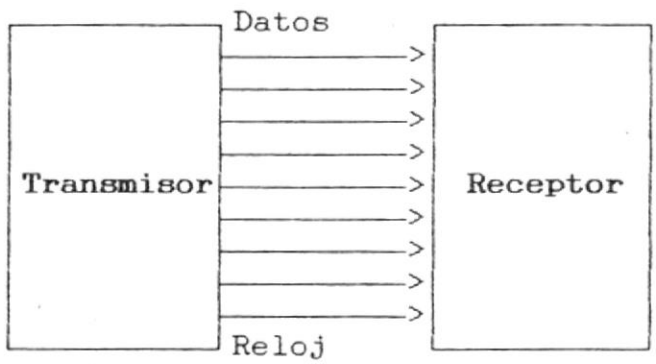


Fig 1.1 Transmisión Paralela

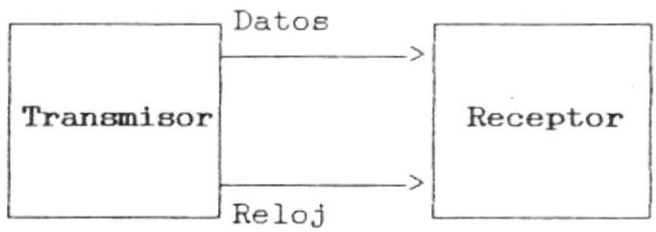


Fig 1.2 Transmisión Serial

El método paralelo es utilizado para transmisiones a gran velocidad entre dos sistemas, no obstante, cuando la distancia entre ambos aumenta, el costo de la línea y el de los amplificadores de transmisión y recepción puede llegar a crecer de forma tal que, desde el punto de vista económico, sea preferible utilizar un sistema de comunicación serial.

Por otra parte, como se verá más adelante, los sistemas de comunicación serial han alcanzado un alto grado de estandarización. Existen normas universalmente aceptadas que fijan completamente todos los detalles de la comunicación, incluyendo aspectos mecánicos (tipo de conector y distribución de señales en los pines del mismo), aspectos eléctricos (niveles y formas de las señales) y aspectos lógicos (sistemas de codificación y sincronización, y descripción de todos los circuitos de datos, control y temporizado).

Por último, un tercer campo en que se utilizan sistemas de manipulación de datos en serie es el de los controladores de unidades de almacenamiento de informaciones digitales sobre soportes magnéticos (discos, cassettes y diskettes). En ellos se graban y se leen los datos en forma serial, presentándose problemas comunes con los sistemas de comunicación serial.

1.3 SINCRONIZACION

Cabría ahora preguntarse: ¿ Cómo hace el computador o el equipo terminal para reconocer cuándo un caracter termina y

otro empieza?, es decir, ¿cómo esta sincronizada la máquina?.

La sincronización de los datos requiere dos fases:

- Sincronización de bits; el terminal debe detectar el instante preciso para chequear si está recibiendo un 1 o un 0.

- Sincronización de caracteres; el terminal necesita conocer qué grupo de bits forman un caracter.

1.3.1 SINCRONIZACION POR BITS

La sincronización de bits implica conocer cuando enviar y muestrear la información y está relacionada con la tasa a la cual se transmiten los bits.

Tanto el equipo transmisor como el receptor, deberán enviar y recibir datos a velocidades idénticas.

El reloj del dispositivo de recepción muestrea los datos entrantes a intervalos correspondientes con los ciclos de reloj de transmisión. Si la línea posee un nivel alto, el reloj de recepción al tomar la muestra detecta un 1, si es bajo, detecta un 0.

1.3.2 SINCRONIZACION POR CARACTER

Puesto que los equipos de transmisión y recepción se encuentran ya en capacidad de reconocer los mismos bits, deben ahora poder identificar qué grupo de bits forman un caracter.

Si los caracteres enviados, por ejemplo, tienen siete bits

cada uno, la máquina necesitará saber cuáles siete bits componen un caracter.

Existen dos métodos conocidos de transmisión de datos en serie: Transmisión Asincrónica y Transmisión Sincrónica. En cada uno de estos métodos, los caracteres están compuestos en forma diferente. El primero usa un método simple; el segundo, uno más complejo.

- **Asincrónico:** Cada caracter va señalizado mediante dos bits, uno al inicio, bit de arranque, y otro al final, bit de parada. El bit de arranque tiene una duración de 1 tiempo de bit, mientras que el bit de parada puede ser de longitud variable, esto es, 1, 1.5, o 2 tiempos de bit. Estos bits permiten reconocer las fronteras de los caracteres.

- **Sincrónico:** Cada mensaje o bloque de transmisión va precedido por unos caracteres de sincronismo.

Para obtener la sincronización de caracter pueden utilizarse diversos sistemas, unos se basan en la utilización de líneas adicionales a las de datos para enviar impulsos que indican el inicio de un bloque de caracteres. Tal impulso identifica el primer bit del primer caracter de un bloque o mensaje, y luego, por contaje de bits y caracteres se determinan todas las fronteras de los datos del bloque.

1.4 CLASES DE ENLACES

En un sistema de comunicación de datos, la línea es el medio

de transmisión entre dos o más equipos terminales. Dependiendo de la aplicación y del equipo disponible, el método de enlace puede ser: Simplex, Semidúplex y Dúplex.

1.4.1 Simplex

Se define como la telecomunicación en una sola dirección. El terminal transmisor solo puede enviar datos a una estación receptora.

1.4.2 Semidúplex

Cuando son capaces de transmitir información en ambos sentidos pero no en forma simultánea.

1.4.3 Dúplex

Cuando son capaces de transmitir simultáneamente información en ambos sentidos.

La codificación de las señales en estos sistemas se hace mediante uno de los siguientes métodos: asíncrono o síncrono.

1.5 EL METODO DE TRANSMISION ASINCRONICA

En este método, la transmisión se controla por bits de inicio y de final que enmarcan cada caracter transmitido, son los denominados bits de inicio y de parada y son utilizados por el terminal receptor para sincronizar su reloj con el del transmisor en cada caracter.

Este método es llamado Asincrónico por cuanto los caracteres

son transmitidos aleatoriamente; esto es, el tiempo entre caracter y caracter puede variar, por ejemplo, cada vez que la persona en el terminal presione una tecla.

Los caracteres pueden ser de cualquier longitud predefinida. Cada caracter sin embargo, debe incluir información que indique la iniciación y terminación del mismo; usualmente un bit de arranque y uno o dos bits de parada

El bit de arranque, dice al receptor, que un caracter está comenzando y que empiece el muestreo de bits de datos. El dispositivo receptor cuenta entonces un número predefinido de bits y los interpreta como un caracter. El bit de parada coloca el sistema en posición de espera hasta que se reciba el siguiente bit de arranque. La figura 1.3 ilustra el método de transmisión asincrónica.

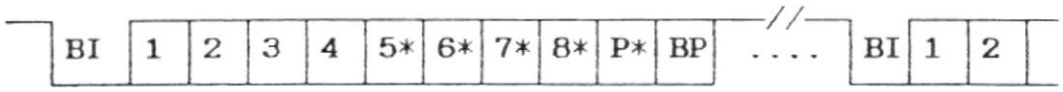


Fig. 1.3.- Formato del patrón serial. Cada caracter es precedido por un bit de inicio (BI) en nivel bajo, el cual sincroniza el reloj del transmisor con el reloj del receptor. Después sigue el bit menos significativo del caracter (de 5, 6, 7, u 8 bits). El signo * indica bits opcionales. Luego un bit de paridad opcional y uno o más bits de parada (BP) en nivel alto, para finalmente quedar la línea en nivel alto hasta el inicio de un nuevo caracter.

La especificación RS404 de la EIA (Asociación de

Industrias Electrónicas) define las caracteríticas del método de tranmisión serial asincrónica.

La transmisión asincrónica se basa en las siguientes reglas:

a) Cuando no se envían datos por la línea, ésta se mantiene en estado 1.

b) Cuando se desea transmitir un caracter, se envía primero el bit de inicio, que pone la línea a cero durante el tiempo de 1 bit.

c) A continuación se envían todos los bits del caracter a transmitir con los intervalos que marca el reloj de transmisión.

d) A continuación del último bit del caracter se envía el bit de parada que hace que la línea se ponga a 1 por lo menos el tiempo de un bit.

Los datos codificados según estas reglas pueden ser detectados fácilmente por el receptor. Para ello deben seguirse los siguientes pasos:

1) Esperar una transición de 1 a 0 en la señal recibida.

2) Activar un reloj de frecuencia igual a la del transmisor.

3) Muestrear la señal recibida al ritmo de este reloj para formar el caracter.

4) Leer de la línea un bit adicional y comprobar si es 1 para

confirmar que no ha habido error de sincronización.

El bit de parada tiene la misión de llevar la línea a estado 1 para que el bit de inicio del próximo caracter provoque la transición del 1 a 0 que permita al receptor sincronizar el siguiente caracter. Este bit también sirve para dar tiempo a que el receptor acepte el dato recibido. De todas formas, actualmente se utilizan registros enclavadores que almacenan el dato recibido mientras el receptor está recibiendo el siguiente, de forma que el microprocesador dispone del tiempo necesario para recoger el caracter recibido.

El método de transmisión serial asincrónica presenta las siguientes ventajas:

1) Permite enviar caracteres a ritmos variables ya que cada uno de ellos lleva incorporada la información de sincronismo.

2) Existen circuitos integrados de bajo costo, las UART que simplifican enormemente la realización de sistemas de entrada/salida en este formato

3) Es un método de comunicaciones estándar entre computadores y terminales de pantalla, impresoras lentas, etc.

Entre sus inconvenientes se puede citar, como más importante su ineficiencia, ya que cada caracter va acompañado de por lo menos dos bits de sincronización que no contienen información útil. Asumiendo caracters de 8 bits, es necesario enviar por

la línea 10 bits para enviar cada caracter, es decir, sólo un 80 % de la información transmitida es válida, el resto es de control.

Los microprocesadores pueden ser programados para realizar las operaciones de serialización y deserialización, sincronización de caracteres o bloques, generación y detección de paridad y generación de los caracteres de control de errores de los bloques. Si bién éste es el sistema más económico para realizar las operaciones de comunicación serie, presenta una serie de inconvenientes:

- No permite alcanzar velocidades de transmisión mayores a los 4800 baudios.
- Consume todo el tiempo de proceso del microprocesador, impidiéndole la realización de otras tareas.

La utilización de integrados controladores permite superar los dos anteriores inconvenientes con un costo muy reducido.

Los receptores y transmisores serie son los encargados de convertir las informaciones paralelo en serie y viceversa, de acuerdo con algún sistema estándar de comunicación. Presentan por un lado una interfase paralela clásica y por el otro lado una línea serial. Son circuitos integrados LSI que pueden llegar a ser muy complejos. Los componentes más conocidos son los UART para comunicación asincrónica.

Los sistemas de comunicación serial tienen a su disposición

un conjunto de recomendaciones elaboradas por asociaciones e institutos de normalización (ISO, EIA, CCITT..) que especifican con precisión todas las características del sistema de comunicaciones.

Las normas para comunicaciones serie están clasificadas por niveles. La norma más ampliamente aceptada es la EIA RS-232C que define las características funcionales, eléctricas, y mecánicas de la interfase entre un terminal y un equipo de comunicaciones (por ejemplo un modem).

Para poder realizar la conexión de terminales a computadores o impresoras a cualquier equipo digital, se necesita una interfase. La interfase es la conexión del equipo a la línea de comunicaciones.

Como mínimo, esta interfase necesita tener hilos comunes para la transferencia de datos y señales que permitan controlar la dirección del flujo de datos.

Si solamente un fabricante hiciera todos los equipos de comunicación de datos, la interfase no sería problema; sin embargo este no es el caso. El mercado lo comparten cientos de fabricantes que construyen cantidades de piezas y equipos inimaginables, aumentando cada vez la posibilidad de confusión.

1.6 LA INTERFASE RS-232C

En los comienzos de la historia de los computadores, se dijo

que por lo menos debería existir una interfase común construída en el interior de todos los equipos.

La Asociación de Industrias Electrónicas (EIA) estableció entonces un conjunto de normas a tener en cuenta, que se conoce como interfase EIA RS-232C, cuyo equivalente internacional es el CCITT V.24. Básicamente, la interfase RS-232C garantiza:

- Que los niveles de voltaje y de señal sean compatibles.
- Que los conectores de la interfase puedan ser acoplados de tal manera que correspondan los pines y sus respectivas conexiones.
- Que cierta información de control suministrada por un equipo pueda ser interpretada por el otro.

La interfase RS-232C incluye 25 pines aunque pocos sistemas los usan todos. En la tabla 1.1 se indican los pines más comúnmente usados y sus funciones.

Cuando se hable de interfases es importante distinguir entre datos enviados por el terminal (DTE) y datos enviados por el equipo de comunicación (DCE). DTE hace referencia al terminal, el cual puede ser un dispositivo de entrada de datos, una impresora o un computador, mientras que DCE se refiere al equipo de comunicación de datos, el modem.

El hecho de utilizar la misma interfase RS-232C en ambos equipos, no garantiza por sí mismo el que los dispositivos

trabajen una vez conectados. Debe asegurarse adicionalmente que las señales correctas estarán presentes en los pines adecuados.

SEÑAL	PIN	DESCRIPCION
GND	1	Conexión al chasis metálico del terminal. (Protective Ground)
TXD	2	Datos salientes, desde el punto de vista del terminal. (Transmited Data)
RXD	3	Datos entrantes, desde el punto de vista del terminal. (Received Data)
RTS	4	Activado por el terminal para decir al modem que se prepare a recibir y retransmitir los datos del terminal. (Request To Send)
CTS	5	Activado por el modem para decir al terminal que está listo para recibir y retransmitir los datos. (Clear To Send)
DSR	6	Activado por el modem para decir al terminal que el modem está operacional. (Data Set Ready)
GND	7	Retorno para las señales del bus. (Signal Ground)
CD	8	Activado por el modem para decir al terminal que ha hecho contacto con el modem remoto y detecta la portadora. (Received Line Signal Detector)
DTR	20	Activado por el terminal para decir al modem que el terminal está operacional. (Data Terminal Ready)

Tabla 1.1 Señales comunmente usadas en RS-232C.

Una interfase RS-232C queda descrita por la definición formal

del estándar. El estándar EIA, RS-232C es una interfase entre equipos terminales de datos (DTE), por ejemplo, un computador o un terminal, y equipos de comunicación de datos (DCE, modem por ejemplo), utilizando un intercambio de datos binarios en serie. Como establece la propia definición, RS-232C es únicamente un "estándar" que formula un conjunto de reglas para el intercambio de datos entre máquinas comerciales como pueden ser los terminales, impresoras, e incluso los computadores personales.

1.7 LA RED TELEX

Telex es un sistema mundial público, conectado, de teleimpresión. Funciona a razón de 66 palabras por minuto (50 bits por segundo) y usa una clave de 5 bits por caracter que se describe más adelante. Cualquier teleimpresor del sistema puede comunicarse con cualquier otro teleimpresor del mismo país y las máquinas Telex pueden conectarse internacionalmente sin necesidad de conversiones de velocidad o de clave. Algunos países permiten que las instalaciones Telex se usen para otras formas de comunicación para transmisión de datos. Cada llamada de Telex se factura sobre una base de tiempo y distancia. Cuando se envía un mensaje a un teleimpresor sin operador, ésta se conectará automáticamente, imprimirá el mensaje y luego se desconectará. Los usuarios de Telex pueden hacer conecciones internacionales a otros países.

TELEX proviene de TELEPRINTER EXCHANGE y se lo define como un

servicio telegráfico para uso particular o privado.

El servicio de Télex establecido por IETEL, constituye un sistema de intercomunicación telegráfica que permite a los usuarios intercambiar mensajes escritos en forma automática, semiautomática o manual mediante el uso de terminales conectados a la red Telex, tales como: teleimpresores y equipos de computación con interfaz para servicio télex.

1.7.1 CARACTERISTICAS TECNICAS PARA UN EQUIPO TELEIMPRESOR

Desde el punto de vista de la calidad de transmisión de los conjuntos terminales arrítmicos se utiliza el Alfabeto Telegráfico Internacional CCITT número 2. Se recomiendan además las siguientes características:

a) Velocidad de modulación 50, 75, 100 baudios.
b) Distorsión de transmisión : Menor del 2 por ciento. La diferencia entre la velocidad de modulación real y la velocidad nominal no debe exceder +/- 0.75 por ciento.
c) Margen de recepción: Mayor al 45 por ciento
d) Duración nominal del ciclo de transmisión de 7.5 unidades como mínimo y la del elemento de parada de 1.5
d) Modo de funcionamiento: Dúplex o semidúplex (y opcionalmente punto a punto).
e) Corriente simple o doble; elegible 2 a 4 hilos.
   - Corriente simple:
     Voltaje telegráfico 120 VDC +/- 10 por ciento
     Corriente de línea en estado de reposo 0 a 5 mA

Corriente de línea en estado de trabajo 40mA, ajustable.

- Corriente doble:

Voltaje telegráfico +/- 48, +/-60 (y opcionalmente 80 VDC con tolerancia de +/- 10 %

Corriente de línea +/- 20mA con tolerancia de +/- 10 %

La línea debe estar aislada del equipo de forma optoelectrónica.

Se ha normalizado un texto internacional para la medición del margen de un teleimpresor, cuyo texto es:

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

También se ha tomado en consideración que los equipos de teleimpresión puedan recibir comunicaciones sin la intervención del operador, ventaja que es aprovechada por los abonados de télex al servicio internacional, lo cual es conveniente para el abonado llamante para verificar la identidad de su corresponsal en el caso de no obtener alguna respuesta, razón por la que se suministra a estos aparatos de un distintivo ANSWERBACK o INDICATIVO, constituido por una serie de 20 señales a saber: letras o cifras, retorno de carro, avance de línea, 16 señales a elección del abonado y de las administraciones, y nuevamente letras; teniendo presente que siempre esas 16 posiciones deben ser ocupadas por lo menos con letras con el objeto de que el abonado solicitante tenga la posibilidad de observar claramente el distintivo solicitado.

El indicativo de llamada se compone de dos partes: El número del abonado y el indicativo literal; el número del abonado consta de 5 dígitos, el primero es el prefijo que identifica a la región, 2 para región 1 y 4 para la región 2.

Ej. 43509: Número de abonado en Guayaquil.

Donde: 4 es el prefijo de la región 2

3509 es el número de abonado en la central

y ESPOL ED es el indicativo literal del abonado

Las letras ED después de la identificación del abonado significan ECUADOR. Para llamar al departamento de información de télex del IETEL marcamos el 104+. Para el exterior y dependiendo del sistema al que se conecte, los números pueden ser de 5 dígitos o más y normalmente van precedidos por el código del país. Por ejemplo, el código de Estados Unidos puede ser 023 o 025, dependiendo de la red a la que esté conectado el abonado en ese país.

Sea cual fuere el número del abonado, siempre se debe conocer de antemano su ANSWERBACK o INDICATIVO, de otra manera podemos enviar mensajes a destinatarios equivocados o simplemente perder los mensajes dentro del sistema.

1.7.2 EL CODIGO BAUDOT

Los principios básicos que han permitido el desarrollo de las redes telegráficas junto a las telefónicas e independientes de las mismas radican especialmente en la transmisión de impulsos, (señal binaria) que se caracteriza mediante los

símbolos 0 y 1, equivalentes eléctricamente a los estados de corriente, positivos o negativos, formando 32 (2 exp 5) combinaciones diferentes, correspondientes al alfabeto CCITT número 2, a través de líneas o radioenlaces.

Ordinariamente en telegrafía se ha usado una clave de cinco bits, como se ve en la tabla 1.3. Esta clave restringiría el número de caracteres a 32 bits, si se usara normalmente. Sin embargo, se usan caracteres de cambio de letras y de cambio de figuras para ensanchar esa gama. Cuando se envía un caracter de cambio de figuras, todos los caracteres que siguen son mayúsculas, hasta que se envía un caracter de cambio de letras. Del mismo modo, los caracteres que siguen a un cambio de letras son letras, hasta que se envíe un caracter de cambio de figuras. los caracteres de cambio de letras y de cambio de figuras deben reconocerse en cada caso.

Código BAUDOT						Minúsculas (Letras)	Mayúsculas (Cifras)
	BIT						
Dec.	4	3	2	1	0		
24	1	1	0	0	0	A	-
19	1	0	0	1	1	B	?
14	0	1	1	1	0	C	:
18	1	0	0	1	0	D	Quién es UD?
16	1	0	0	0	0	E	3
22	1	0	1	1	0	F	Nota 1
11	0	1	0	1	1	G	Nota 1
5	0	0	1	0	1	H	Nota 1
12	0	1	1	0	0	I	8
26	1	1	0	1	0	J	Bell
30	1	1	1	1	0	K	(
9	0	1	0	0	1	L	)

Código BAUDOT						Minúsculas (Letras)	Mayúsculas (Cifras)
7	0	0	1	1	1	M	.
6	0	0	1	1	0	N	,
3	0	0	0	1	1	O	9
13	0	1	1	0	1	P	0
29	1	1	1	0	1	Q	1
10	0	1	0	1	0	R	4
20	1	0	1	0	0	S	'
1	0	0	0	0	1	T	5
28	1	1	1	0	0	U	7
15	0	1	1	1	1	V	=
25	1	1	0	0	1	W	2
23	1	0	1	1	1	X	/
21	1	0	1	0	1	Y	6
17	1	0	0	0	1	Z	+
0	0	0	0	0	0	Espacio en blanco	
31	1	1	1	1	1	Cambio de letras	
27	1	1	0	1	1	Cambio de números	
4	0	0	1	0	0	Estado	
2	0	0	0	1	0	Retorno del carro	
8	0	1	0	0	0	Avance de líneas	

Tabla 1.3: La clave telegráfica BAUDOT (CCITT, alfabeto internacional telegráfico normal número 2).

Nota 1: No se asigna internacionalmente. Se pone a diposición de cada país para uso interno.

1.7.3 ABREVIATURAS USUALES EN LA OPERACION TELEX

Las siguientes abreviaturas son de uso universal en la operación télex, por lo cual también pueden ser utilizadas en los contactos internacionales.

ABS Abonado ausente, instalación desconectada.

ANUL Anulen.

BK Corte, suspenda transmisión.

CFM Ruégole confirme-confirmo.

CRV ¿ Recibe bién ? - ¿ Recibo bién ?.

DER Abonado en reparación.

DF Esté en comunicación con el abonado solicitado.

EEE Error.

XXXX Error.

GA Puede transmitir. Adelante.

INF No puede obtenerse el abonado por el momemto. Llame al servicio de información.

JFE Instalación cerrada por día festivo.

MNS Minutos.

MOM Momento.

MUT Mutilado.

NA No se admiten llamadas para este abonado.

NC No hay circuitos disponibles para llegar hasta el abonado deseado; congestión.

NCH El número del abonado llamado ha sido cambiado.

NI Sin identificación de línea disponible.

NP El solicitado ya no es abonado.

NR Indique su número de llamada-mi número de llamada es...

N17 El indicativo del abonado llamado no funciona.

OCC La máquina del abonado está ocupada.

OK De acuerdo. ¿ Está de acuerdo ?

P (o cifra 0) Interrumpida la transmisión.

RAP Volveré a llamarle.

RPT Repita-repito

SVP Haga el favor

TAX ¿ Cuál es la tasa ?-La tasa es de...

TEST Ruégole envíe un mensaje de prueba.

THRU Está en comunicación con una posición télex.

WRU ¿ Quién llama ?

TPR Teleimpresor.

El próximo capítulo describe las características del computador personal; proporciona una breve descripción del microprocesador 8088, sus registros internos y la forma de cómo se direcciona a la memoria del computador.

CAPITULO DOS

DESCRIPCION GENERAL DE LA FAMILIA IBM PC

2.1 INTRODUCCION

Desde el punto de vista del programador, todos los miembros de la familia PC contienen un microprocesador, integrados de memoria, y varios circuitos integrados inteligentes o programables. Todos los componentes del circuito principal, que necesita el computador para trabajar, están localizados en la placa principal del sistema; otra parte muy importante se encuentra en las tarjetas de expansión, y pueden conectarse en la placa del sistema.

La placa principal del sistema contiene el microprocesador (bien el 8088 o el 80286) junto con al menos 64 Kbytes de memoria, algunos programas internos en ROM, tales como el BASIC y la ROM-BIOS, y varios integrados de soporte muy importantes. Estos integrados controlan dispositivos externos, como la pantalla o el controlador del disco, y realizan algunas otras ayudas al microprocesador durante la realización de sus tareas.

En esta sección describiremos los integrados principales y algunas de sus características principales.

2.1 EL MICROPROCESADOR 8088

El 8088 es el microprocesador de 16 bits que controla a los computadores personales estándar IBM, incluyendo al original PC, el XT, el PC portátil y el PCjr. Es la unidad central del proceso (CPU) del computador. Cada bit de datos que entra o sale del computador pasa a través de la CPU para ser procesado, o redirigido.

El 8088 controla las operaciones básicas del computador, envía y recibe señales de control, direcciones de memoria y datos, de una parte del computador a otra, a través de la red electrónica de interconecciones llamada **bus**. A lo largo del **bus** se encuentran los puertos de entrada y salida (E/S), las diversas memorias y los integrados de soporte. Los datos pasan a través de estos puertos de E/S y viajan a la CPU a las otras partes del computador, o viceversa.

Dentro del 8088 existen 14 registros, que le proporcionan un área de trabajo adecuada para poder realizar la transferencia de los datos y su procesamiento. Estos registros internos constituyen una memoria interna de 128 bytes de tamaño, y pueden almacenar datos, direcciones de memoria, punteros de instrucción e indicadores de estado de control. A través de estos registros, el 8088 puede acceder a más de un millón de bytes y 64k puertos de E/S.

2.2 LOS CONTROLADORES DE PERIFERICOS

El microprocesador no puede controlar el computador entero

sin recibir ayuda. Para ello delega algunas de las funciones de control a otros integrados, quedando la CPU libre para atender a su propio trabajo. Estos integrados de soporte se responsabilizan de procesos, como por ejemplo, controlar el flujo de información dentro del computador, como lo hacen los controladores de interrupciones y de acceso directo a memoria (DMA), o el flujo de información con un dispositivo externo que está unido al computador, tales como una pantalla de video, o una unidad de disco. A menudo, estos dispositivos se encuentran en tarjetas separadas que se insertan en los conectores de expansión de los PC.

Algunos de los integrados de soporte del IBM PC son programables, lo que significa que se pueden manipular para que realicen tareas especiales. A continuación describiremos las funciones que desempeñan cada uno.

El Controlador de Interrupciones 8259: El cual supervisa la realización de las interrupciones. Las interrupciones son señales enviadas a la CPU por la circuitería para requerir su atención, o responder a alguna acción.

El Controlador de Acceso Directo a Memoria o DMA 8237A: Su propósito principal es permitir al controlador del disco, leer o escribir datos sin involucrar al microprocesador. Como las operaciones de E/S desde el disco son relativamente lentas, el DMA puede aumentar un poco las prestaciones del computador.

El Generador de Reloj 8284A: Proporciona las señales de temporización multifase que son necesarias para el funcionamiento del microprocesador y de los periféricos.

El Interfaz de Periféricos Programable 8255: Se utiliza para conectar los dispositivos periféricos del computador al bus. La información que se envía a dispositivos tales como el parlante o la información recibida desde el teclado, viaja a través de los puertos de E/S por medio de este integrado.

El Reloj Programable 8253: Es un reloj de propósito múltiple y un controlador que puede generar tres tiempos de retardo exactos, bajo control de programa. Un contador 0 que genera una interrupción de programa 8h, el contador 1 que se utiliza para el refrescamiento de la memoria RAM y el contador 2 que controla el puerto de cassette o el parlante..

2.3 COMO SE COMUNICA EL 8088.

Para comprender cómo se puede usar el Lenguaje Ensamblador, es necesario comprender cómo procesa la información el 8088 y cómo trabaja con el resto del computador. El objetivo de discusión de este capítulo es estudiar cómo se comunica el 8088 con la parte del computador.

El Lenguaje Ensamblador del 8088, como el resto de los otros lenguajes, está formado por un conjunto de códigos de instrucciones simbólicas. En el interior del 8088 estos códigos y los datos que están asociados con ellos están representados en forma binaria, con objeto de que puedan

residir en la memoria y se muevan a través de la circuitería electrónica para realizar ciertas tareas específicas. A esta representación se la denomina Lenguaje de Máquina.

Las operaciones que ejecutan las instrucciones del 8088 se pueden clasificar en unas cuantas categorías: las que hacen operaciones simples, como son las cuatro funciones aritméticas con números enteros de 8 y 16 bits; las que se emplean para mover datos de un lado a otro; las que son capaces de manipular bits individualmente, utilizando métodos de desplazamiento; las que suelen comprobar valores y tomar desiciones lógicas en función de los resultados, y por último las que pueden interaccionar con la circuitería de su entorno. El tamaño de cada instrucción varía de uno a seis bytes.

El 8088 interacciona con la circuitería que le rodea de tres formas: Accede a la memoria de forma directa o indirecta, a través de los puertos y mediante señales denominadas interrupciones.

La memoria se utiliza leyendo o escribiendo valores que están almacenados en localizaciones de memoria que se identifican mediante direcciones numéricas. A estas localizaciones de memoria se puede acceder de dos formas: directamente a través de un controlador de acceso directo a memoria (DMA), o indirectamente a través de los registros internos del 8088.

El controlador del disco y las puertas de comunicaciones en

serie pueden acceder directamente a la memoria a través del controlador DMA. El resto de los dispositivos transfiere los datos (a y de) la memoria a través de los registros del 8088.

Los puertos son el modo general de comunicación que emplea el 8088 con el resto de los componentes de la circuitería de la memoria. Como en el caso de las localizaciones de memoria, los puertos se identifican mediante un número, y los datos se pueden leer, o escribir, en ellos. La asignación de un puerto es único y característico del diseño de un modelo de computador particular.

Generalmente, todos los miembros de la familia IBM PC utilizan las mismas especificaciones de puerto, con algunas variaciones entre los diferentes modelos.

Las interrupciones son la forma empleada por la circuitería exterior al 8088 para informar que algo (como, por ejemplo, la pulsación de una tecla) ha ocurrido y que, por lo tanto, requiere que se realice alguna acción. Aunque las interrupciones son esenciales para la interacción del 8088 con su entorno, el concepto de interrupción se utiliza también con otros propósitos. Por ejemplo, el sistema BIOS, o el Sistema Operativo de Disco (DOS), pueden producir interrupciones mediante programas para requerir y ejecutar subroutinas especiales de servicio.

2.4 DIRECCIONAMIENTO DE MEMORIA EN EL 8088

El 8088 es un microprocesador de 16 bits, y por lo tanto no

puede trabajar directamente con valores mayores de 16 bits; por consiguiente, el valor decimal más grande que puede representar es el 65 535, o 64Kb. Teóricamente, esto significa que el 8088 sólo podría acceder a 64K direcciones de memoria. Sin embargo, el 8088 puede direccionar hasta 1 MB de memoria. Esto es posible porque el 8088 usa un esquema de direccionamiento de 20 bits, ampliando el rango de posiciones de memoria desde 2exp16 (65 535) a 2exp20 (1 048 576). Pero como el 8088 está limitado por su capacidad de procesamiento de 16 bits, para acceder a direcciones de 20 bits necesita utilizar un procesamiento de direccionamiento que se adapte al formato propio de 16 bits.

El 8088 divide el espacio de memoria direccionable en un numero arbitrario de segmentos, cada uno de los cuales contiene como máximo 64Kbytes. Cada segmento empieza en una localización cuya dirección es divisible por 16 bytes, y que se conoce como **dirección del segmento o párrafo del segmento**. Para acceder a los bytes, se emplea una dirección adicional llamada **desplazamiento** u **offset**, que apunta a la posición de un byte exacto dentro del segmento de 64K designado por la dirección del segmento. Las direcciones internas siempre son relativas al comienzo del segmento, por lo que también se denominan **direcciones relativas**.

Las direcciones se crean y se manipulan combinando la dirección del segmento de 16 bits y una dirección relativa de 16 bits. La dirección de segmento se trata como si estuviera

desplazada a la izquierda cuatro bits. Cuando se añade la dirección relativa, se obtiene una dirección completa de 20 bits, tal como aparece en la figura 2-1. A las dos palabras de 16 bits usualmente se las denomina **dirección segmentada**, o también **vector**, cuando se hace referencia a las interrupciones.

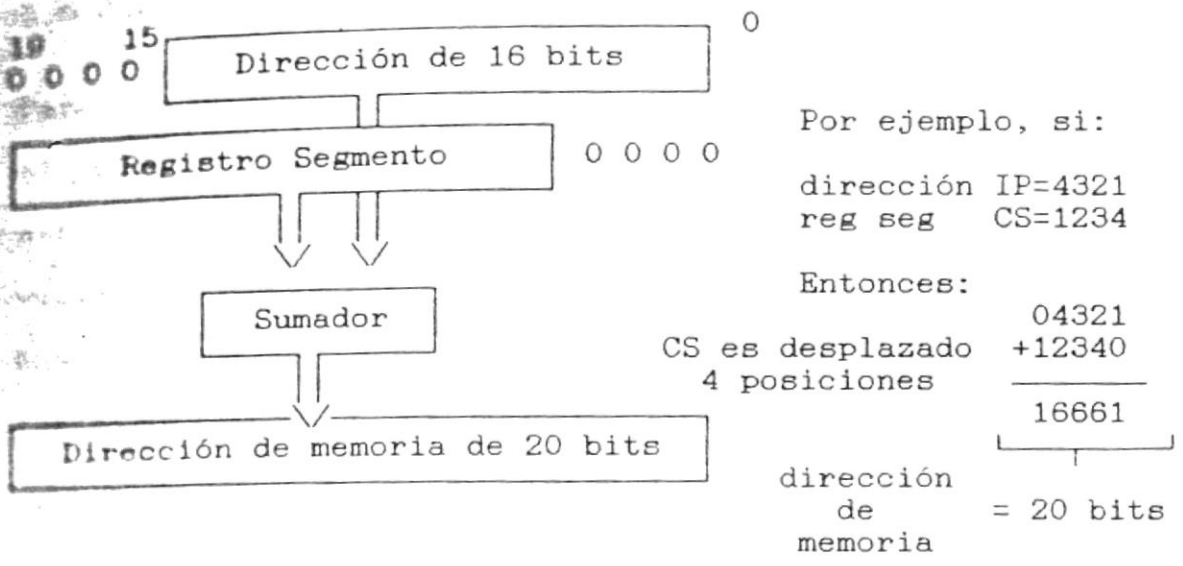


Fig. 2-1. Como las direcciones de memoria son formadas por el 8088.

El hecho de que la parte de segmento de una dirección segmentada se desplaze a la izquierda cuatro bits, es la razón por la cual la parte segmento sólo puede apuntar a las direcciones de memoria reales que son múltiplo de 16, y por ello es necesaria la dirección relativa para definir la posición precisa dentro del segmento. Las direcciones relativas se escriben con valores hexadecimales de 4 dígitos. Cuando se suman estos dos números, forman otro número hexadecimal de 5 dígitos, que es una dirección de 20 bits.

Por ejemplo, si escogemos una dirección de segmento hexadecimal tal como 1234 y lo multiplicamos por 16, obtenemos 12340. Luego, si le añadimos la dirección relativa del byte que estamos buscando, tal como 4321, obtenemos un resultado hexadecimal de 5 dígitos, tal como se puede observar en la figura 2-1.

2.5 JUEGO DE REGISTROS DEL 8088

Para los propósitos de programación en Lenguaje Ensamblador, el microprocesador 8088 puede ser visto simplemente como un set de registros cuyos contenidos pueden ser modificados de varias maneras por medio de instrucciones de Lenguaje Ensamblador. La figura 2-2 muestra el set completo de los registros del 8088.

REGISTROS DE TRABAJO

AX	AH	AL	Acumulador
BX	BH	BL	Base
CX	CH	CL	Contador
DX	DH	DL	Dato

REGISTROS RELATIVOS

IP		Puntero de instruccines
SP		Puntero de pila
BP		Puntero base
SI		Indice orígen
DI		Indice destino

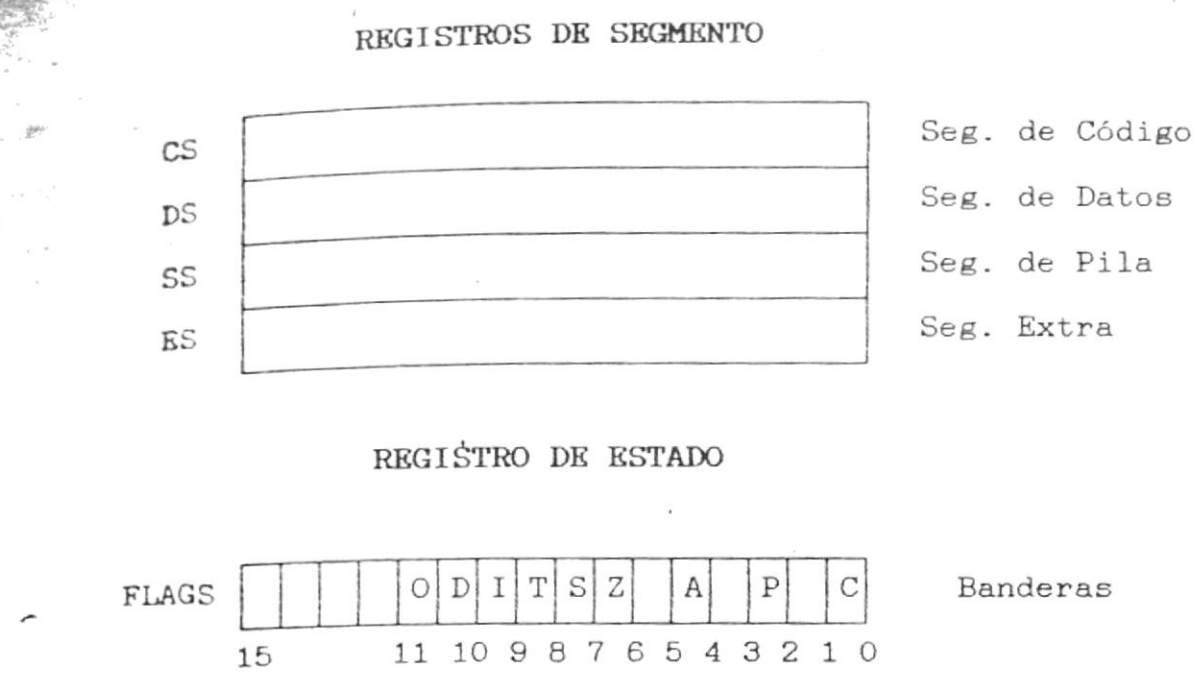


Fig. 2-2. Juego de registros del 8088

Existen catorce registros en total, cada uno de 16 bits. Cuatro **registros de trabajo** que se utilizan temporalmente por los programas para almacenar resultados intermedios y los operandos que necesitan las operaciones aritméticas y lógicas Cuatro **registros de segmento** que almacenan las direcciones iniciales de ciertos segmentos en la memoria. Cinco **registros relativos** que almacenan las direcciones relativas que se emplean con las direcciones de segmento para indicar dónde están situados los datos en la memoria. Finalmente, hay un **registro de estado**, formado por nueve bits que se emplean para registrar la información del estado del 8088 y de las operaciones de control del 8088.

Los registros de trabajo o de datos son los más utilizados. Cuando un computador está procesando datos, gran parte del tiempo el microprocesador lo gasta en traer y llevar datos a

la memoria. Este tiempo de acceso se puede reducir sustancialmente en muchas ocasiones, guardando en el interior del 8088 los operandos más usados así como los resultados obtenidos. Con este propósito se utilizan los registros de trabajo.

Los registros de trabajo se conocen como AX, BX, CX y DX. Cada uno de ellos se puede a su vez subdividir y direccionar de forma separada, como si fueran dos registros de 8 bits. Los registros de mayor orden se conocen como AH, BH, CH y DH, y los registros de orden bajo se conocen como AL, BL, CL y DL.

Los registros de trabajo se emplean la mayor parte de las veces como áreas temporales de trabajo, en particular para realizar operaciones aritméticas. La suma y la resta se pueden hacer en memoria sin utilizar los registros, pero éstos se emplean menos tiempo y, por lo tanto, las operaciones se realizan de forma más rápida.

Aunque estos registros se encuentran disponibles para realizar cualquier tipo de trabajo, cada uno tiene asignado algunos usos especiales.

- El registro **AX** es un acumulador, y es el registro principal para realizar las operaciones aritméticas o también operaciones de entrada/salida por los puertos de comunicación.

- El registro **BX** (base) se utiliza a menudo para apuntar el

comienzo de una tabla en memoria. Se puede utilizar también para almacenar la parte relativa de una dirección segmentada.

- El registro **CX** (contador) se utiliza como un contador de repetición para control de **lazos** . Por ejemplo, la instrucción LOOP utiliza CX para almacenar el contador que indica el número de iteraciones del lazo. Ninguno de los otros registros puede realizar esta función.

- El registro **DX** se utiliza para almacenar datos de 16 bits con propósitos generales.

Los registros relativos se utilizan para localizar un byte en concreto, dentro de un segmento específico de 64K. El registro IP, llamado **puntero de instrucción** o **contador del programa**, indica cuál es la dirección donde se encuentra situada dentro del segmento de código la instrucción que se va a ejecutar. Se utiliza junto con el registro CS. Existen además dos registros que se denominan **registros de pila**, que están íntimamente ligados a la pila, que no es más que una zona de la memoria donde el 8088 guarda información de direcciones y de datos, que necesita recordar para su utilización posterior. Los dos registros que restan se denominan **registros de índice** y se utilizan para señalar la situación de los operandos que se van a utilizar dentro del segmento de datos.

Los programas no tienen acceso directo al registro IP, pero hay un conjunto de intrucciones, tales como JMP y CALL, que

cambian el contenido a o desde la pila.

Los registros de pila, denominados puntero de pila, SP y **puntero** de base BP, proporcionan direcciones relativas **dentro** del segmento de pila. SP proporciona la situación de **la** cabecera de la pila y es análogo al registro IP. El **registro** BP se utiliza para indicar, partiendo de la posición **inicial** de la pila, donde se encuentra determinada **información**. Este registro es particularmente utilizado para **crear** rutinas de interfaz en lenguaje ensamblador.

Los registros de índice, denominados el índice de fuente SI **y el** índice de destino DI, se utilizan normalmente con otro **registro** (AX, BX, CX o DX) o con ciertas instrucciones, que **proporcionan** la dirección relativa del lugar inicial de un **campo** de datos que esté comprendido en el interior de un **segmento** de datos. Se utilizan muy a menudo cuando se **transfieren** grandes listas de datos. Las instrucciones de **cadenas** que utilizan SI y DI transfieren los bytes de uno en otro. Luego los registros SI y DI incrementan sus valores **automáticamente**, de tal forma que cuando termina cada **transferencia** no se tiene que añadir uno cada vez para mover **el siguiente** byte de información.

Como acabamos de ver, la dirección completa de una dirección de memoria consiste en la dirección de un segmento de 64Kb y la dirección interna de un segmento. Los cuatro registros, llamados CS, DS, SS y ES se utilizan para identificar cuatro segmentos de 64KB específicos de memoria.

- El registro CS localiza el **segmento de código**; es decir, la dirección a partir de la cual empieza el programa que está siendo ejecutado.

- El registro DS localiza el **segmento de datos** que es la dirección donde empieza el área de memoria que almacena los datos que se están utilizando.

- El registro SS localiza el **segmento de pila**, que es la dirección donde empieza un lugar de trabajo temporal que almacena parámetros y direcciones que utiliza el programa que está activo.

- El registro ES, apunta hacia un **segmento extra**, que se utiliza normalmente como suplemento del segmento de datos, con objeto de que se pueda utilizar más de 64KB de memoria para almacenamiento. También se utiliza para transferir datos entre segmentos.

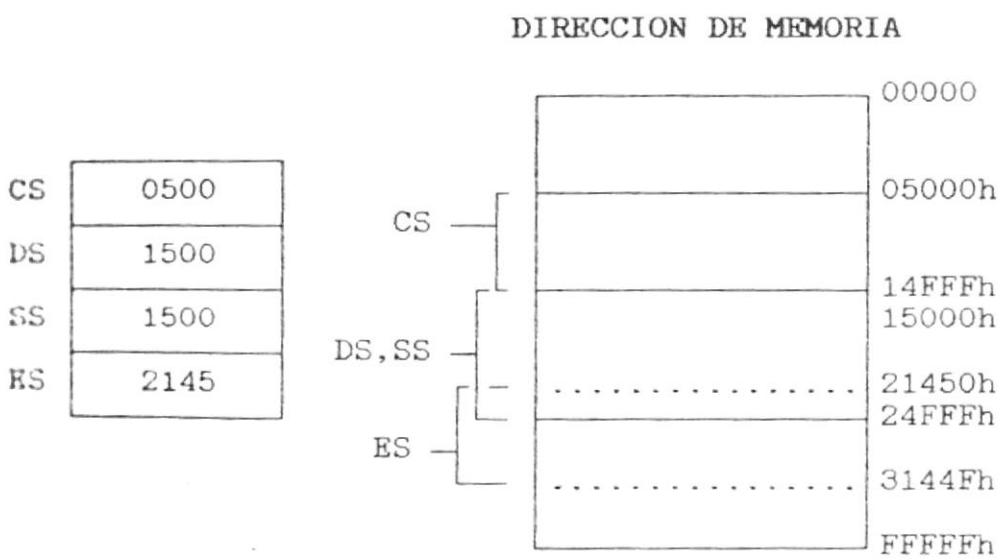


Fig. 2-3. Ejemplo de como los segmentos de memoria son direccionados usando los registros de segmentos.

Es común que los cuatro segmentos se solapen o sean idénticos; también es común que sólo se utilice una parte del segmento de 64KB; por ejemplo, un programa puede necesitar tan sólo 16KB de un segmento de 64KB. La figura 2-3 ilustra cómo se puede distribuir la memoria.

Todas las instrucciones del 8088 que utilizan la memoria también emplean implícitamente el registro de segmento apropiado para que la operación se realice. Por ejemplo, la instrucción MOV, cuando actúa con datos, utiliza el registro DS. La instrucción JMP, que afecta al flujo de un programa, utiliza automáticamente el registro CS.

El **registro de estado** es realmente una colección de bits individuales de control, denominados **indicadores** o **flags** que están organizados en forma de registro, de manera que o bien pueden ser guardados y recuperados como si constituyeren un dato, o bien los indicadores se modifican y se inspeccionan como elementos independientes; esta segunda técnica suele ser la más normal.

Existen nueve indicadores de 1 bit en el registro de estado de 16 bits. Estos indicadores se pueden dividir de forma lógica en dos grupos: seis indicadores de estado, que se utilizan para registrar la información del estado del procesador (que señalan normalmente lo que ocurre con una comparación y operación aritmética), y tres indicadores de control, que dirigen a algunas de las instrucciones del 8088. Los términos que aparecen en la figura 2-4 son los más

comunes.

INDICADORES DEL REGISTRO DE ESTADO DEL 8088

Indicadores de Estado		Indicadores de Control	
CF	Arrastre	DF	Dirección
OF	Desbordamiento	IF	Interrupción
ZF	Cero	TF	Detención de programa
SF	Signo		
FP	Paridad		
AF	Arrastre auxiliar		

Fig. 2-4. Los indicadores de estado y de control del registro de estado del 8088.

Hemos visto que la memoria se direcciona siempre mediante la combinación de un valor de la dirección del segmento y de un valor relativo, y que la parte del segmento de una dirección siempre viene de uno de los cuatro registros de segmento. La parte relativa puede venir de una combinación de uno, dos o tres de las siguientes fuentes.

- Un valor relativo que aparece en la propia instrucción.
- Un registro, usualmente AX, BX, CX o DX.
- Un registro de índice, SI o DI.

Es importante conocer que se pueden aplicar un conjunto de reglas cuando se utilizan los registros, y es esencial tenerlas en cuenta cuando se escriban rutinas en lenguaje ensamblador. Las reglas y convenciones de uso varían con las

circunstancias y con el lenguaje de programación; desafortunadamente no existen reglas de oro que se apliquen siempre, pero aquí daremos algunas, que se pueden aplicar en la mayoría de las ocasiones.

En una rutina de interfaz construida en lenguaje ensamblador, hay tres formas generales de utilizar los registros: algunos registros pueden alterarse libremente; otros pueden cambiarse pero deben volver a su situación inicial al final de la rutina, y el resto no puede cambiarse nunca.

Generalmente, los registros de trabajo (AX a DX) pueden variar libremente, sin causar ningún daño al programa de llamada. Algunas reglas particulares se deben aplicar a los cuatro registros de segmento (CS, DS, SS y ES). El registro CS no debe ser cambiado nunca directamente, aunque se puede cambiar indirectamente a través de llamadas a subrutinas. El registro DS puede variar, pero normalmente es necesario restablecer su valor inicial después. El valor original del registro SS es necesario preservarlo siempre que se hagan cambios en el registro. Normalmente, las subrutinas continúan usando la pila a la que apuntaba SS. El cambio de valor de SS puede interferir con el uso del puntero base (BP) para los parámetros de acceso. El registro ES se puede variar normalmente a voluntad.

El puntero de instrucciones IP no debe de variar directamnete, como en el caso del registro CS ya que los cambios se producen de forma automática.

El puntero de pila SP puede cambiar, pero, normalmente, todos los cambios de este registro se producen como resultado del uso indirecto de la pila.

El puntero base BP se cambia normalmente para conseguir el acceso directo algunos parámetros, y, a menudo, al finalizar se restablece su contenido.

Los registros de índice (SI y DI) pueden ser alterados libremente cuando sea necesario.

En el registro de estado, los indicadores de estado pueden variar rutinariamente. Algunos de los indicadores de estado se utilizan ocasionalmente para señalar que se ha producido algún resultado; por tanto, su inicialización puede ser importante.

2.6 USO DE LOS PUERTOS DE ENTRADA/SALIDA

El 8088 se comunica y controla algunas partes del computador a través de los puertos de entrada y salida (E/S).Los puertos de E/S son zonas físicas a través de las cuales pasa información a o desde un dispositivo de E/S, como por ejemplo un teclado o una impresora.

Cada puerto se identifica mediante un número de 16 bits, que puede estar comprendido entre 0 y 65535. La CPU envía datos , o información de control a una puerta determinada, especificando su número, y la puerta entonces responde pasando datos, o información de su estado, a la CPU.

De la misma forma que cuando accede a la memoria, la CPU utiliza los buses de datos y direcciones como los conductos de comunicación con los puertos. Para acceder a un puerto, la CPU envía una señal por el bus de control que notifica a todos los dispositivos de E/S que la dirección colocada en el bus de direcciones es de un puerto; posteriormente, el dispositivo correspondiente al puerto activado responde.

El número de puerto direcciona una posición de memoria, que es parte de los dispositivos de E/S, pero no de la memoria principal. Se usan instrucciones especiales de E/S para señalar el acceso a un puerto y enviar información en ambos sentidos a los dispositivos de E/S. Algunos dispositivos de E/S, tales como los controladores de video, también usan las direcciones de la memoria principal, además de sus puertos de E/S, y hace que la CPU las tome como una parte de la memoria principal. Esto se conoce como E/S mapeada en memoria. Generalmente, los dispositivos mapeados en memoria son más fáciles de programar, porque nos permiten utilizar las instrucciones inflexibles y limitadas de E/S del conjunto de instruccines del 8088.

2.7 INTERRUPCIONES DEL 8088

Siempre que un dispositivo de comunicación, o un programa, necesita la atención de la CPU, envía al microprocesador una señal, o instrucción, llamada **interrupción**, identificando el tipo de tarea particular que quiere realizar. Cuando el microcomputador recibe la señal de interrupción, generalmente

para todas las otras actividades, activa una rutina almacenada en memoria, llamada **controlador de interrupciones**, que está asociada al número particular de la interrupción. Después de que la rutina de tratamiento de interrupción ha realizado su tarea, la actividad del computador continúa donde estaba cuando ocurrió la interrupción.

Existen tres categorías principales de interrupciones. En primer lugar, hay interrupciones generadas por la circuitería del computador en respuesta a algún acontecimiento. Estas interrupciones están manejadas por el integrado controlador de interrupciones 8259, el cual las prioriza por orden de importancia antes de enviarlas a la CPU para que actúe. En segundo lugar, hay interrupciones que son generadas por la CPU como resultado de alguna actuación inusual producida por el programa, por ejemplo, una división por cero. Y en tercer lugar, hay interrupciones generadas deliberadamente por los programas como una forma de invocar subroutinas almacenadas en RAM o ROM. Estas interrupciones, a menudo llamadas interrupciones por software, son comunes en las rutinas de servicio de la ROM-BIOS y del DOS.

Siempre es posible modificar el programa de las rutinas de tratamiento de las interrupciones, o escribir otras si la aplicación lo requiere.

Además de estas interrupciones, existe también un tipo especial de interrupción, llamada interrupción no enmascarable (**NMI**), que se utiliza para solicitar la atención

inmediata de la CPU. A menudo indica que se ha producido una emergencia, como por ejemplo, una caída de voltaje, o un error de memoria. Cuando se envía una NMI, como tiene la más alta prioridad, la CPU la atiende antes que al resto de las interrupciones.

Siempre que se genera una interrupción, el causante de la misma, no necesita conocer la dirección de memoria de la rutina encargada de su tratamiento, sólo necesita conocer el número asociado a la interrupción. Este número apunta a una tabla de memoria almacenada en las posiciones de memoria más baja, que contiene la dirección segmentada de la subroutina encargada del tratamiento de la interrupción. A la dirección inicial de la rutina encargada del tratamiento de la interrupción se la denomina **vector de la interrupción**, y la tabla se llama **tabla de vectores de interrupción.**

Las interrupciones guardan automáticamente los valores contenidos en los registros CS e IP. De esta forma, el computador puede volver al sitio donde estaba trabajando cuando se produjo la interrupción. Además, el proceso de la interrupción guarda el registro de estado en la pila y borra el indicador de interrupción IF, previniendo temporalmente la actuación de otras interrupciones. Normalmente, una subroutina de tratamiento de interrupción termina tan pronto como es posible, normalmente con la ejecución de unas pocas instrucciones.

Existe una instrucción especial de retorno de la

interrupción, **IRET**, que realiza esta función; es análoga a la instrucción **RET** usada con llamadas o subroutinas. IRET restablece también los indicadores y los registros CS e IP.

Es bastante común enlazar subroutinas hechas en ensamblador a programas, o a lenguajes de programación, de tal forma que se pueda realizar el acceso a las rutinas de servicio del DOS y del BIOS y reforzar la eficacia de un programa. Para relizar este tipo de rutinas, y en especial, las que llaman a rutinas de servicio del DOS o del BIOS, es necesario programar en Lenguaje Ensamblador. Aunque para la mayor parte de las necesidades, estos interfaces consisten en llamadas a subroutinas sencillas, o en llamadas de interrupción mediante la instrucción **INT**. Sólo los lenguajes de programación más avanzados incluyen la posibilidad de creación de rutinas que permitan el tratamiento de interrupciones y el uso de la instrucción IRET.

En el siguiente capítulo veremos cómo podemos usar las diferentes instrucciones del Lenguaje Ensamblador y la manera de podemos accesar a los puertos del computador personal.

CAPITULO TRES

PROGRAMACION EN LENGUAJE ENSAMBLADOR DEL MICROPROCESADOR 8088

3.1 INTRODUCCION

El Lenguaje Ensamblador es una forma de lenguaje de máquina usado por los computadores para ejecutar los diferentes programas. Conociendo su uso antes que la circuitería del computador nos ayuda a descubrir cómo opera el computador y el porqué la circuitería ha sido diseñada de esa manera. El Lenguaje Ensamblador proporciona la facilidad de hacer que un computador realice literalmente cualquier cosa que éste es físicamente capás de hacer. Esto es esencial cuando se desean diseñar programas para controlar los dispositivos de entrada/salida del PC, para agregar nuevas interfaces de E/S, y en general para realizar tareas que están fuera del alcance de los lenguajes de alto nivel tales como el BASIC o el PASCAL. Un dispositivo de E/S sin el programa de control necesario resulta totalmente inservible.

Este capítulo introduce los conceptos básicos y la terminología utilizada por el Lenguaje Ensamblador en términos del set de instrucciones del microprocesador 8088 y muestra cómo un computador moderno funciona internamente al

realizar sus operaciones básicas. Este también muestra cómo usar el IBM Macro Ensamblador para escribir programas sencillos.

3.2 FORMATO DE LAS DECLARACIONES EN LENGUAJE ENSAMBLADOR

Cada línea de un programa en Lenguaje Ensamblador debe seguir un formato definido, para que el Ensamblador pueda interpretar corréctamente lo que se quiere hacer. El formato es muy simple. Cada línea de código ensamblado está dividida en cuatro áreas o campos como se muestra en la figura 3.1.

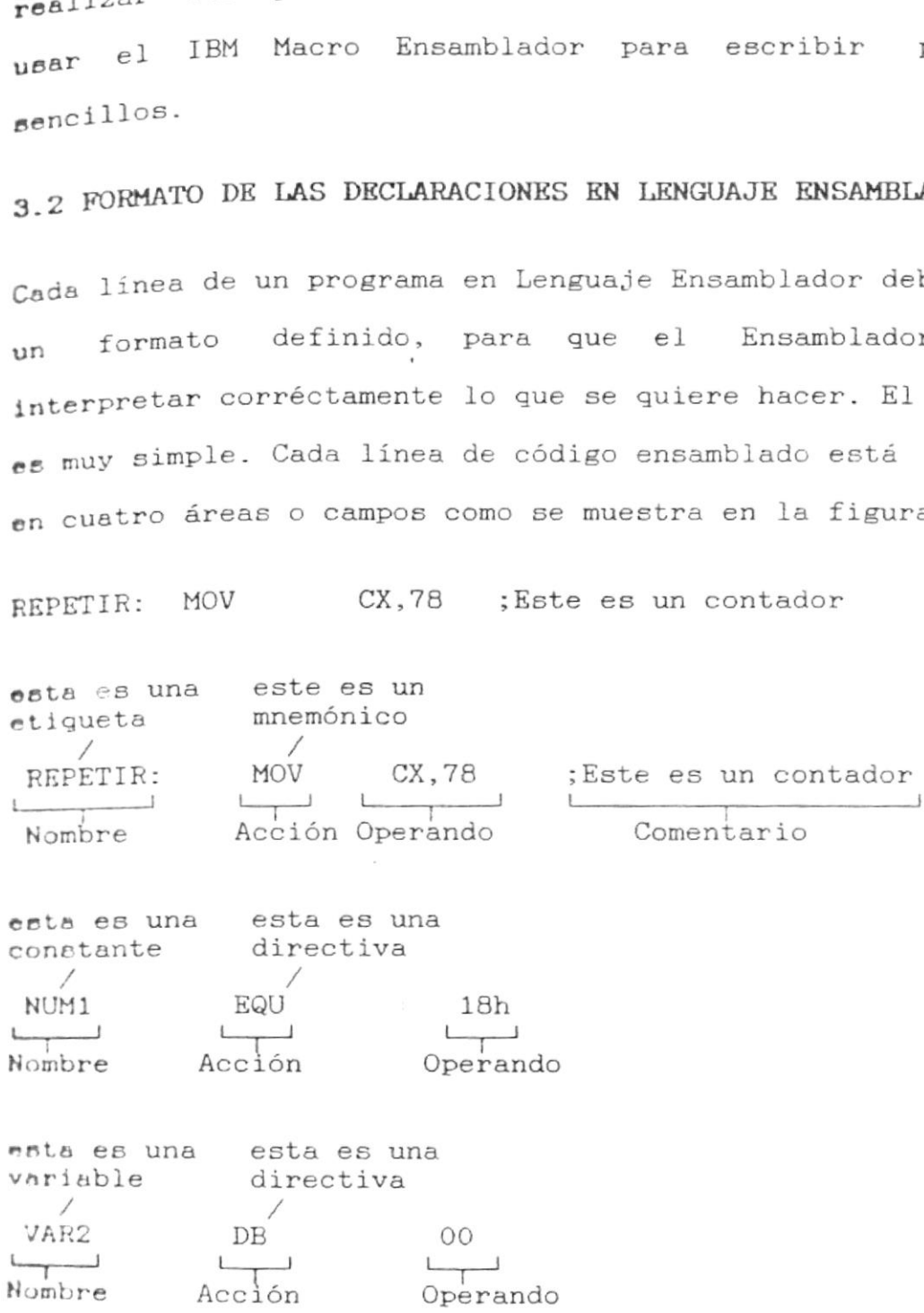


Fig. 3.1 Formato de las declaraciones en Lenguaje Ensamblador

El primer campo es el "Campo de nombres". En el primer ejemplo, este campo contiene una etiqueta llamada REPETIR. El

Ensamblador reconoce REPETIR como una etiqueta por el simple hecho de que ésta finaliza con dos puntos (:). Las etiquetas se usan para dar un nombre simbólico a la dirección donde se almacenan las instrucciones, y se usan como referencia para las instrucciones que se deseen ejecutar más de una vez durante la ejecución de un programa. El campo de nombres puede no llevar etiqueta.

El segundo campo es el "Campo de acción". La palabra en este campo indica al Ensamblador que acción debe realizar. En el ejemplo, este campo contiene la palabra MOV, la cual dice al computador mover algún dato. Una instrucción de máquina contenida en este campo es llamada "mnemónico". En lugar de una instrucción, este campo también puede contener una directiva del Ensamblador, la cual es una instrucción para el Ensamblador, y no una parte del programa del computador.

El tercer campo es el "Campo de operandos". La expresión contenida en este campo proporciona al Ensamblador algún dato adicional que éste necesita para realizar la acción. En el ejemplo, éste campo contiene dos operandos: CX y 78. Esto indica que el dato origen es el número 78, y que el destino para este dato es el registro CX. Luego, el número 78 es almacenado en el registro CX. Cuando dos operandos están presentes, estos deben separarse por una coma (,) sin espacios intermedios. Un mnemónico dado puede requerir 0, 1 o 2 operandos. Por ejemplo, la instrucción RET no requiere

operandos.

El campo final es el "Campo de comentarios". Este campo debe comenzar con punto y coma (;). En cualquier línea del programa fuente que el Ensamblador encuentre un ";" éste interpretará que todo lo que sigue a la línea es un comentario. Los comentarios solo sirven como referencia para el programador, por lo tanto el Ensamblador los ignora. Algunas líneas de un programa en Lenguaje Ensamblador no contienen instrucciones de máquina. En lugar de esto, contienen directivas para el Ensamblador. He aqui algunos ejemplos:

```
TITLE     PROGRAMA DE PRUEBA
PAGE      60,96
NUM1      EQU     18H
VAR1      DW      1234H
VAR2      DB      00
END
```

TITLE, PAGE, EQU, DW, DB y END son directivas del Ensamblador a menudo llamadas pseudo-operaciones. El Ensamblador determina si una línea de un programa es una instrución de máquina o una directiva del Ensamblador, viendo la primera palabra de la línea. Si esta palabra finaliza con ":", es una etiqueta para una localización del programa; si es uno de los mnemónicos del 8088 (tales como MOV o JMP), la línea es una instrucción de máquina. Si la primera palabra no es una etiqueta o un mnemónico, el Ensamblador chequea si

ésta es una directiva legal (tales como TITLE o PAGE). Si es una de esas palabras reservadas, el resto de la línea es interpretada de acuerdo a lo requerido por la directiva.

Por ejemplo, la directiva TITLE indica que todo lo que sigue a ésta sobre la línea será el titulo o encabezamiento de cada página de impresión del listado del programa en Lenguaje Ensamblador. Este listado muestra tanto el programa en el Lenguaje Ensamblador (mnemónicos) como el lenguaje de máquina producido por el Ensamblador. La directiva PAGE dice que los números que siguen a la línea serán usados como el número de líneas por página y el máximo número de caracteres por línea en el listado del programa en lenguaje Ensamblador. La directiva END indica la finalización del programa.

Otra posibilidad es que la primera palabra sobre una línea no sea una etiqueta, un mnemónico o una directiva. El Ensamblador entonces interpreta la palabra como un nombre y chequea la siguiente palabra de la línea para ver que clase de nombre es este. En los ejemplos dados, NUM1, VAR1 y VAR2 son nombres. La directiva **EQU** indica que el nombre que precede a ésta es una constante cuyo valor es el numero que sigue a EQU. En el ejemplo dado, la constante NUM1 simplemente será interpretada como el número 18H cada vez que aparezca en el programa.

La directiva DW (Define Word) significa que el nombre que la precede es una variable y que el Ensamblador reservará 2 bytes de memoria (una palabra) para esa variable. Además, el

Ensamblador inicialmente llenará esta palabra de memoria con el valor que sigue a DW. Por ejemplo, VAR1 es una variable cuyo valor inicial es 1234H. La directiva DB (Define Byte), es igual a DW excepto que solo un byte es reservado para la variable en memoria. En el ejemplo, VAR2 es una variable de un byte cuyo valor inicial es 00.

3.3 MODELO PARA UN PROGRAMA EN LENGUAJE ENSAMBLADOR

Ciertas directivas del Ensamblador deben incluirse aún en el programa más simple, de tal manera que el programa pueda ser ensamblado correctamente. Esta sección presenta un programa modelo cuyo formato puede ser usado para muchos programas en Lenguaje Ensamblador y explica los detalles necesarios. La figura 3.1 muestra el esquema de un simple programa en Lenguaje Ensamblador.

La primera línea indica el título del programa; esta línea principalmente es para información. La línea debe comenzar con la palabra reservada **TITLE** o el Ensamblador podría dar un mensaje de error. La última línea del programa debe tener la declaración **END**.

El programa está dividido en tres secciones, cada una de las cuales se denomina un Segmento del Programa. Los segmentos comienzan con un nombre del segmento, seguido por la palabra reservada **SEGMENT** y finalizan con una línea en la cual se repite el nombre del segmento seguido de la palabra reservada **ENDS**. Un segmento es un bloque de código de máquina o

simplemente datos que pueden ser direccionados por un valor contenido en uno de los registros de segmento. La declaración **ASSUME** indica qué registro es asignado a cada segmento. Así, en el programa dado, el registro CS apunta a CSEG (este segmento contiene el programa) y DS apunta al segmento DSEG donde están almacenadas las variables, DSEG. Normalmente, el 8088 direcciona a las variables por medio del registro DS. No es necesario dar un valor ASSUME al registro SS porque el DOS automáticamente lo fija al segmento de pila.

```
TITLE    PROGRAMA MODELO

DSEG     SEGMENT                      ;Usualmente,todas
MSG      DB  'ESTE ES UN MENSAJE$'    ;variables van en el
DSEG     ENDS                         ;segmento de datos

SSEG     SEGMENT STACK                ;El DOS automática-
         DW 80 DUP (?)                ;mente fija una
SSEG     ENDS                         ;pila en el segmen-
                                      ;to de pila.
CSEG     SEGMENT
         ASSUME  CS:CSEG,DS:DSEG
FUENTE   PROC    FAR                  ;Reserva la direc-
         PUSH    DS                   ;ción inicial del
         SUB     AX,AX                ;programa en la
         PUSH    AX                   ;pila
         MOV     AX,DSEG              ;Entonces fija DS
         MOV     DS,AX                ;en el segmento de
                                      ;datos.
         MOV     DX,OFFSET MSG        ;Este es el inicio
         MOV     AH,09                ;del programa
         INT     21H                  ;principal. Solo
                                      ;tiene tres lineas
         RET                          ;de longitud !
FUENTE   ENDP                         ;Retorno hacia el
CSEG     ENDS                         ;DOS

         END
```

Fig. 3.1 Modelo para un simple programa en Lenguaje Ensamblador

Además, se tiene el segmento de pila el cual se ha denominado SSEG. Para definir el segmento de pila es necesario usar la directiva SEGMENT seguida de la palabra reservada **STACK**, lo cual indicará al Ensamblador que SSEG contiene la pila. Cuando un programa es cargado a memoria para su ejecución, el DOS automáticamente posiciona los registros SS:SP al final del segmento de pila. Para el contenido del segmento de pila se ha indicado DW 80 DUP (?). Esto significa que el Ensamblador reservará 80 palabras de memoria sin importar el contenido de esas localidades de memoria.

El código del programa está contenido en el segmento de código, el mismo que se ha denominado CSEG. Aquí, el programa es escrito como una simple subrouţina la misma que el Macro Ensamblador IBM llama un PROCEDIMIENTO. La subroutina debe comenzar con una línea indicando un nombre para la misma ("FUENTE" en el ejemplo), seguido por la palabra reservada **PROC** y de la palabra **FAR**. El final de la subroutina se indica por una línea que repite el nombre de la subroutina seguido por la palabra reservada **ENDP**.

Si se desea usar este modelo para otros programas, únicamente deben sustituirse las tres líneas de código: MOV DX,OFFSET MSG; MOV AH,09 e INT 21H por los nuevos programas, además de la linea MSG DB 'ESTE ES UN MENSAJE$' con algunas variables o datos que se necesiten para los programas. Los programas en Lenguaje Ensamblador pueden escribirse usando un Editor de Texto y luego almacenarce en el disco como un archivo.

Supongamos que tenemos un archivo que contiene el programa "EDITOR.ASM". Entonces se debe cargar el programa Ensamblador y ensamblar este programa ingresando: **MASM EDITOR.ASM;**. Este ensamblará el programa y almacenará el lenguaje de máquina producido sobre el disco como un archivo llamado EDITOR.OBJ. Luego de esto, se debe correr el programa LINK (que viene suministrado por el DOS), ingresando: **LINK EDITOR.OBJ;**.El programa LINK toma el código de máquina y lo convierte a una forma que pueda ser cargado y ejecutado diréctamente desde el DOS. El programa LINK, produce un archivo en el disco llamado EDITOR.EXE, el cual puede ser ejecutado desde el DOS simplemente ingresando EDITOR.

Ahora veremos porque razón el programa es escrito como una subroutina y qué función desempeñan las primeras líneas del segmento de código. Lo que se está tratando de hacer es "arreglar" las cosas para que cuando se ejecute la última línea del programa, el computador "salte" hacia el DOS y espere por comandos adicionales. Esto podría realizarse simplemente con una instrucción de salto. Desafortunadamente el lugar de retorno al DOS está en el segmento DS:0, donde DS es el contenido del registro DS cuando comienza la ejecucion del programa, y el Ensamblador simplemente no nos permite "saltar" hacia una localización absoluta de memoria. Así, ¿ cómo llegamos a DS:0 ?. La manera más simple es guardar la dirección DS:0 en la pila y entonces retornar desde ésta. Por lo tanto, las primeras tres líneas del programa de código almacenan el contenido de DS en la pila y entonces almacenan

el valor 0000. La última línea del programa es una instrucción de retorno lejano. El Ensamblador reconoce cuando es un retorno lejano, en el cual tanto CS como IP son modificados, chequeando la palabra FAR colocada después de la palabra reservada PROC. La instrucción de retorno lejano toma las dos últimas palabras puestas en la pila y las encaja los registros CS e IP; de esta manera, CS:IP ahora apuntan a DS:0, que es el punto apropiado de entrada al DOS.

En la cuarta y quinta línea del programa el registro DS es modificado, de modo que apunte a DSEG. Esto se realiza para tener acceso hacia las variables contenidas en el segmento DSEG. Cuando el programa es cargado en la memoria, el DOS se encarga de la inicialización de SS:SP para apuntar al segmento de pila y fija CS:IP al inicio del programa para su ejecución, de modo que no es necesario inicializar los registros de segmento CS o SS.

3.4 INSTRUCCIONES DE MOVIMIENTO DE DATOS

El resto de este capítulo presenta un detalle del conjunto de instrucciones disponibles para el 8088. El primer tipo de instrucción a tratarse es la instrucción **MOV**. La forma de esta instrucción es "**MOV destino,origen**". Todas las instrucciones del 8088 usan este formato en el cual el origen de los datos es dado en último lugar. Una intrucción relacionada con la instrucción MOV es la instrucción de intercambio "**XCHG destino,origen**". Esta instrucción intercambia ya sea el contenido de dos registros o de un

registro y una localización de memoria. Con XCHG, ambos operandos son origen y destino.

La tabla 3.1 resume la forma de las instrucciones de movimiento de datos. Los datos pueden ser movidos desde un registro hacia otro registro o localización de memoria, pero no entre dos localizaciones de memoria. Para conseguir esto, se requieren dos instrucciones MOV, tales como MOV AL,mem1 seguido de MOV mem2,AL.

Un valor también puede ser cargado diréctamente hacia un registro o localización de memoria como se muestra en la tercera forma de instrucción MOV. Un movimiento directo de memoria a memoria puede realizarse en una sola instrucción usando las instrucciones de cadenas MOVS discutidas en la sección 3.11.

Instrucciones de Movimiento de datos

MOV	reg,reg/mem	XCHG	reg,reg/mem
MOV	reg/mem,reg		
MOV	reg/mem,num		
LEA	reg16,mem		

Instrucciones de Entrada/Salida

IN	ac,puerto	OUT	puerto,ac
IN	ac,DX	OUT	DX,ac

donde "ac" es AX o AL, y "puerto" es la dirección de un puerto de entrada/salida

Tabla 3.1 Instrucciones de movimiento de datos y de entrada/salida.

Una instrucción de movimiento de datos de propósito especial y fácil de operar es la instrucción de traslado **XLAT**. Esta instrucción sustituye el valor en AL con un nuevo valor tomado de una lista o tabla de valores en memoria. La instrucción asume que el registro BX apunta al inicio de esta tabla, es decir, BX contiene la dirección del primer valor en la tabla. El valor en AL determina qué valor de la tabla será usado. Por ejemplo, si AL contiene el valor 3, entonces el tercer valor de la tabla direccionada por BX será colocado en AL mediante la instrucción XLAT. En vista de que AL es de 8 bits, la longitud máxima de la tabla puede ser de 256 bytes así como 256 (FF Hex) el máximo valor que puede almacenarse en AL.

El uso más frecuente para la instruccion XLAT es pasar valores de un tipo de código a otro. Un ejemplo es la conversión de un número binario de 4 bits a un valor hexadecimal en código ASCII. Si el número de 4 bits está en el registro AL, esta conversión puede efectuarse con una tabla y dos instrucciones:

```
    MOV     BX,OFFSET TABLA
    XLAT
    ...

TABLA       DB '0123456789ABCDEF'
```

Las instrucciones **IN** (Input) y **OUT** (Output) son un tipo especial de instrucciones de movimiemto de datos que leen o

envían datos desde o hacia un puerto de entrada o salida. Los puertos de entrada y salida son parte de una circuitería especial usada por el computador para comunicarse con dispositivos externos. Desde el punto de vista de programación, estos puertos son simplemente un tipo especial de localizaciones de memoria que pueden ser accesadas solo por medio de las intrucciones IN y OUT.

Existen dos tipos de instrucción IN. El primer tipo tiene la forma IN AL,DX. Esta instrucción coloca en el registro AL el dato disponible en la dirección del puerto de entrada contenida en DX. Esta instrucción no es de propósito general como lo es la instrucción MOV. El destino únicamente puede ser AL (o AX) y la dirección del origen debe estar contenida en DX. La instrucción IN BL,CX no es permitida. Si se usa IN AX,DX el resultado es equivalente a ejecutar lo siguiente:

```
INC     DX
IN      AL,DX
MOV     AH,AL
DEC     DX
IN      AL,DX
```

El segundo tipo de instrucciones IN está más limitado. Este tipo tiene la forma IN AL,puerto; donde "puerto" es la dirección del puerto de entrada (un número entre 0 y FF). Por lo tanto, únicamente los primeros 256 puertos de entrada/salida pueden ser accesados por este tipo de instrucción.

Las instrucciones de salida tienen la misma forma de las instrucciones de entrada y trabajan de la misma manera excepto que el valor en AL es enviado hacia el puerto. Por ejemplo, podemos usar: OUT 37,AL o MOV DX,37 seguida de OUT DX,AL para enviar el contenido de AL al puerto de salida 37. El contenido de AL no es afectado por una instruccion OUT.

3.6 MODOS DE DIRECCIONAMIENTO

Veamos ahora el significado de "mem" en la tabla 3.1. Mem es una dirección de memoria, pero existen varias maneras de especificar una dirección de memoria. Existen 24 distintas posibilidades. Afortunadamente muchas de ellas son variaciones de las formas básicas. Saber como usar los modos de direccionamiento y como utilizarlos es uno de los aspectos mas difíciles del Lenguaje Ensamblador. Entonces procederemos a estudiarlos cuidadosamente con algunos ejemplos.

Para empezar, en Lenguaje Ensamblador no podemos referirnos a la memoria por una dirección especifica. La razón es que usualmente las direcciones absolutas son desconocidas durante la elaboración de un programa. El computador determina la dirección absoluta únicamente cuando el programa es cargado a la memoria para su ejecución. Además no es necesario para un programador tomar en cuenta direcciones absolutas; simplemente se pueden usar nombres simbólicos para todas las localizaciones de memoria que se deseen utilizar. El Ensamblador se encarga de realizar todos los detalles.

Es posible reservar varias localidades de memoria para almacenar datos que un programa necesite. En Lenguaje Ensamblador, un área de almacenamiento de datos podría ser:

```
version     db      13,10,'Telex PC'
alt_shift   db      00001010b
contador    db      1,0
reloj       db      0,0,58,0,0,58
duracion    dw      07FFh
autor       db      'ESPOL'
```

Como anteriormente fue discutido, los nombres en la columna de la izquierda, se denominan VARIABLES. Por ejemplo, CONTADOR es una variable que contiene el número 1, al inicio del programa y puedeir variando a lo largo del desarrollo del mismo.

Recordemos que el mnemónico DB en el campo de acción se usa para Definición de Byte. Esto significa que los números que aparecen en el resto de la línea serán almacenados en memoria; a un byte por número. Algunos valores de un byte pueden definirse en una sola directiva DB, escribiendo los numeros después de DB y separándolos por comas como se ha realizado en la variable RELOJ. Unicamente pueden ser almacenados en un byte números entre 00 y FFh. Si se desea tener una variable con un mayor rango de valores debemos usar la directiva DW. Definición de palabras, la cual almacena el valor que sigue a ésta como un valor de 2 bytes en memoria.

Una vez que se ha definido una variable, no es necesario conocer su dirección. Cada vez que el programa la necesite, únicamente debe accesarla por medio de su nombre. Cuando el Ensamblador encuentra una directiva DB, el número que la precede será asignado a la siguiente dirección de memoria disponible. Supongamos por ejemplo que cuando la variable VERSION fue encontrada, la siguiente dirección disponible era 1020h. Entonces, el área de datos será:

```
1020 0D,0A,54,65,6C,65,78,20,50,67 version db 13,10,'Telex PC'
102A 0A                            alt_shift db 00001010b
102B 01,00                         contador  db 1,0
102D 00,00,3A,00,00,3A             reloj     db 0,0,58,0,0,58
1033 FF,07                         duracion  dw 07FFh
1035 45,53,50,4F,4C                autor     db 'ESPOL'
```

La columna de la izquierda proporciona las direcciones iniciales correspondientes a cada declaración de Lenguaje Ensamblador, y los números a la derecha de las direcciones son los valores que inicialmente están almacenados en esas direcciones. Las etiquetas VERSION Y AUTOR son manejadas de una manera especial.Cuando el Ensamblador encuentre un DB seguido de comillas, éste interpreta que lo que está entre las comillas es una cadena literal. Es decir, se almacena en memoria el valor de código ASCII para cada caracter de la cadena.

Ahora que conocemos como las variables se almacenan en memoria, discutiremos a fondo los modos de direccionamiento.

Supongamos que a lo largo del programa deseamos almacenar el valor de AL en la variable CONTADOR y que también queremos conocer el contenidode la variable RELOJ. Entonces, podemos almacenar el valor de ALen la variable CONTADOR usando la instrucción MOV WON,AL. Esta declaración significa: colocar el contenido de AL en la localización de memoria CONTADOR. Similarmente, MOV AL,RELOJ colocará el contenido de la localización de memoria RELOJ en el registro AL.

Técnicamente, el modo de direccionamiento hasta aqui usado es llamado **direccionamiento directo**, porque la dirección del dato fuente o destino está dada diréctamente en la instrucción del. Lenguaje de Máquina. La instrucción MOV CONTADOR,AL es idéntica a MOV [102B],AL si conocemos que la variable CONTADOR está en la localización 102B. Podría pensarse que MOV [CONTADOR],AL es equivalente a MOV [102B],AL; pero esto no es cierto. De hecho, MOV [CONTADOR],AL, no es una declaración legal en Lenguaje Ensamblador. El Ensamblador trata los registros o variables sobre una posición igual. Cuando se escribe el nombre de un registro o una variable significa usar el contenido de ese registro o variable. Por ejemplo. MOV AL,3 mueve 3 hacia el registro AL, mientras que MOV CONTADOR,3 mueve 3 a la variable CONTADOR. No es posible usar MOV CONTADOR,RELOJ es decir, movimiento directo de memoria a memoria.

Se pueden realizar expresiones más complejas de variables y números. Por ejemplo, MOV AL,RELOJ+3 mueve el contenido de la

localización de memoria 1030 hacia AL. Cuando el Ensamblador encuentra tales expresiones aritméticas, éste chequea la dirección de 16 bits del nombre de la variable que aparece y entonces realiza la operación indicada para obtener la dirección efectiva. El contenido de esta direcion es usado como la fuente o destino para la instrucción.

En algunas situaciones se desea cargar una dirección hacia un registro en lugar del contenido de esa dirección. En tales circunstancias, debemos usar la palabra reservada **OFFSET**.

Por ejemplo, la declaración MOV DX,OFFSET RELOJ pondrá el valor 102D en el registro DX. Nótese que MOV DX,RELOJ y MOV DX,OFFSET RELOJ pson dos instrucciones totalmente diferentes. La expresión "OFFSET,nombre_de_variable", es una manera simbólica de especificar un número. Así, la instrucción MOV DX,OFFSET RELOJ produce un código de máquina para la instrucción MOV DX,102D. Esta clase de instrucción es un ejemplo del llamado **modo de direccionamiento inmediato**. Existe otra alternativa para usar MOV reg,OFSSET nombre_de_variable y es la instrucción **LEA** (Load Effective Address) discutida al final de esta sección.

Además de los modos de direccionamiento DIRECTO e INMEDIATO, existe otro tipo de direccionamiento muy poderoso disponible para el 8088. Este modo es llamado **direccionamiento indirecto o indexado**. En este modo, un registro contiene la dirección de memoria cuyo contenido es usado el origen o destino. Por ejemplo, supongamos que el registro BX contiene el número

102D: entonces, la instrucción MOV AL,[BX] pondrá el contenido de la localización de memoria 102D en AL.

Los paréntesis entre el registro BX en la instrucción MOV AL,[BX], indican al Ensamblador que el dato a cargarse en AL no es el contenido del registro BX, más bien se trata del contenido de la dirección de memoria contenida en BX. Esta es la razón del porque una instruccion como MOV AL,[BX] es llamada de direccionamiento indirecto. El operando dado es un registro, pero el dato a ser cargado no está en el contenido del registro.

El modo de direccionamiento indirecto puede ser usado únicamente con cuatro registros: BX, BP, SI y DI. Por lo tanto, MOV CX,[SI], MOV [DI],BX y MOV [BX],AL son instrucciones válidas.

En ciertos casos de direccionamiento indirecto es necesario especificar el tamaño del destino (1 o 2 bytes). Es obvio que el destino debe ser de 16 bits si la instrucción es MOV [BX],AX puesto que el dato origen está en un registro de 16 bits. Sin embargo, ¿ cómo podría interpretarse la instrucción MOV [BX],3 ?. En esta situación, el Ensamblador no sabría si BX está apuntando a una cantidad de 8 bits o a una de 16 bits. Para resolver esta situación, el Ensamblador suministra las palabras reservadas **BYTE PTR** (Byte Pointer) y **WORD PTR** (Word Pointer). Ellas son colocadas en frente del destino, de tal manera que el Ensamblador conoce cuando se quiere mover un dato de 8 o 16 bits. Por ejemplo: MOV WORD PTR [BX],3

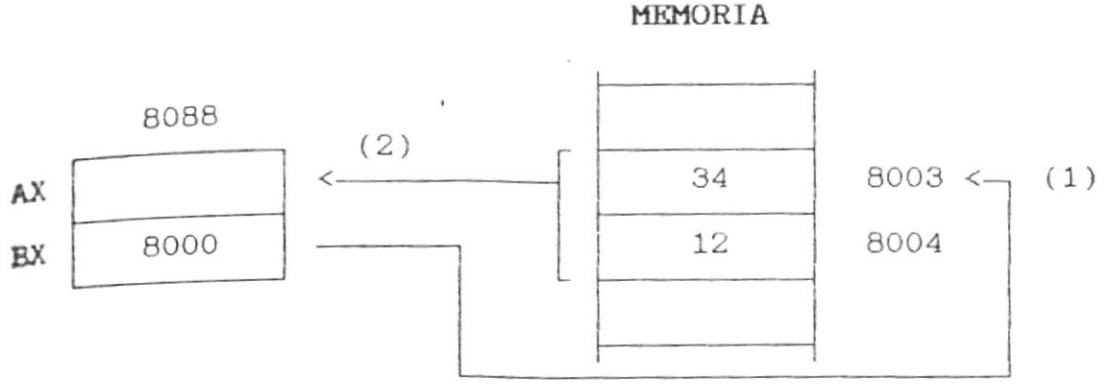
moverá la cantidad 0003 de 16 bits a las localizaciones [BX] y [BX]+1, mientras que MOV BYTE PTR [BX],3 moverá la cantidad 03 de 8 bits hacia la localización [BX]. Normalmente MOV CONTADORN,3 indica un movimiento de 8 bits. Sin embargo, también se puede almacenar una cantidad de 16 bits tal como 1234 en CONTADOR y CONTADOR+1 usando MOV WORD PTR CONTADOR,1234.

Muchas formas complicadas de direccionamiento indirecto o indexado suman números o variable a la cantidad [reg]. Por ejemplo, MOV AL,[BX]+5 y MOV [BX]+RELOJ,AL son instrucciones válidas. Existen otras dos formas equivalentes de expresar la cantidad [reg]+disp, estas son: [reg+disp] y disp[reg]. Por ejemplo, MOV [BX]+RELOJ,AL ; MOV [BX+RELOJ],AL y MOV RELOJ[BX],AL representan la misma intrucción y causan que el Ensamblador genere el mismo código de máquina.

Algo que debemos recordar es que el Ensamblador siempre chequea el operando y realiza todas las operaciones que están indicadas para obtener una dirección efectiva. El contenido de esta dirección efectiva es el dato a ser movido. La figura 3.2 muestra esquemáticamente como opera este modo de direccionamiento. Si BX contiene 8000, entonces MOV AL,[BX]+3 transfiere el contenido de 8000+3= 8003 hacia el registro AL. Similarmente, si BX contiene 05, la instrucción MOV [BX]+RELOJ,AL mueve el contenido de AL hacia la dirección 0005 + 102D = 1032 (si la dirección de la variable RELOJ es 102D). También se puede usar un número y una variable juntos,

por ejemplo, MOV [SI] + 3 + CONTADOR,CL.

La ejecución de una instrucción MOV AX,[BX]+3 causa las siguientes acciones:



1.- Dirección Efectiva: [BX]+3=8003

2.- El contenido de 8003 y 8004 se coloca en AX

3.- Después de la ejecución de la instrucción, AX contiene 1234.

Fig 3.2 Ejecución de una instrucción MOV usando direccionamiento indirecto.

El último y más complejo tipo de direccionamiento es el que usa la suma del contenido de dos registros como una dirección efectiva. Por ejemplo, MOV AX,[BX+SI] es una instrucción valida. Esta instrucción indica al Ensamblador que forme una dirección efectiva de la suma del contenido de los registros BX y SI, y luego mover el contenido de esa dirección efectiva hacia AX. Este modo de direccionamiento está limitado a ser usado solamente los registros BX y/o BP para el primer componente de la suma y SI y/o DI para el segundo componente;

por lo tanto, MOV AL,[SI+DI] no es una instrucción válida.

Todas las diferentes posibilidades de direccionamiento de memoria discutidas están resumidas en la tabla 3.2. Como puede observarse, una dirección efectiva para un modo de direccionamiento indirecto puede obtenerse usando algunas de las siguientes expresiones: [BX o BP], y/o [SI o DI], y/o alguna expresión que contenga un numero de 8 o 16 bits. Así, la estructura general es muy simple aunque existen muchas posibles variaciones.

1. 'num' es un número de 8 0 16 bits, o más generalmente alguna expresión que evalúe a un valor de 8 o 16 bits.

2. 'reg/mem' significa 'reg' (registro) o 'mem' (memoria).

3. 'reg' es uno de los siguientes registros:

AX	SI	AH	AL
BX	DI	BH	BL
CX	BP	CH	CL
DX	CP	DH	DL

'reg 16' es cualquiera de los registros de 16 bits mostrados en las dos primeras columnas

4. 'mem' es una DIRECCION EFECTIVA para cualquiera de las combinaciones formadas de las siguientes columnas:

$$\begin{bmatrix} BX \\ o \\ BP \end{bmatrix} + \begin{bmatrix} SI \\ o \\ DI \end{bmatrix} + disp$$

Donde 'disp' es el nombre de una variable o alguna expresión la cual evalúa a un numero de 8 o 16 bits.

Tabla 3.2 Valores permitidos para los operandos: 'reg','reg 16','reg/mem', 'mem' y 'num'.

En ciertas ocasiones se requiere cargar la misma dirección efectiva hacia un registro en lugar del contenido de la dirección efectiva. Para esta situación, el 8088 incluye la instrucción **LEA** (**L**oad **E**ffective **A**ddress). Por ejemplo, si queremos cargar la dirección de la variable RETARDO hacia el registro BX, podemos usar LEA BX,RETARDO. Esto también puede realizarse con la instrucción MOV BX,OFFSET RETARDO. Los dos métodos realizan la misma cosa excepto que la última técnica usa un byte menos. La utilidad de la instrucción LEA es notoria cuando se desea cargar a un registro una dirección efectiva mas compleja. Por ejemplo, supongamos que queremos cargar la dirección efectiva de [BX+SI] + RETARDO hacia el mismo registro BX. La instrucción LEA BX,[BX + SI]+RETARDO realiza esto, mientras que de otra manera tendríamos que realizar:

```
ADD     BX,SI
ADD     BX,OFFSET RETARDO
```

3.6 SALTOS Y LLAMADOS A SUBROUTINAS

Si toda una computadora se la utiliza para ejecutar una simple lista de instrucciones sin ninguna posibiblidad de modificar sus acciones basadas en resultados previos, esta sería de muy limitada utilidad. Los nuevos microprocesdores, incluyendo el 8088, tienen un buen grupo de instrucciones de transferencias de control. Muchas de ellas son de transferencia de control condicional, en las cuales la decisión de salto a un nuevo lugar en el programa está basado

en una serie de bits de control especiales llamados indicadores de estado o banderas. Estos indicadores y los saltos condicionales son discutidas en la seccion 3.8.

Consideraremos primero dos instrucciones de transferencia de control incondicionales muy útiles: **JMP** y **CALL**. La tabla 3.3 resume las formas permitidas de esas instrucciones.

Instrucciones de Transferencia de Control Condicionales

CALL	etiqueta	RET	
CALL	reg/mem	RET	nn
JMP	etiqueta		
JMP	reg/mem		

donde "etiqueta" es una etiqueta del programa, y nn es un número de 16 bits.

Instrucciones de Pila

PUSH	reg 16/mem	POP	reg 16/mem
PUSHF		POPF	

Instrucciones de Interrupción

INT	n	IRET	
INTO			
CLI			
STI			

donde n es un número de 8 bits

Tabla 3.3 Instrucciones de tranferencia de control incondicionales, pila e instrucciones de interrupción.

Una instrucción JMP es la más simple de las del tipo de transferencia de control. Cuando usamos JMP dirección, la instrucción causa que la CPU coloque la dirección dada en la instrucción, en el registro IP, de tal manera que las siguientes instrucciones a ejecutarse son aquellas que están a partir de esa dirección y no la instrucción que seguía a JMP. Usualmente la dirección es dada en la forma de una etiqueta (un nombre terminado con dos puntos) que comienza en la línea del código ensamblado donde se desea saltar. Por ejemplo, si queremos saltar a un lugar nombrado AQUI escribimos JMP AQUI.

Otro formato de la instruccion JMP es: JMP reg/mem. En este caso, el salto es hacia la dirección contenida o en un registro o en una localización de memoria. Por ejemplo, si el registro BX contiene 1234 entonces JMP BX causa que la ejecución del programa salte a la instrucción contenida apartir de la dirección 1234. Esta forma de instruccion JMP es muy útil para realizar saltos múltiples. Un salto múltiple es construido con una tabla de saltos. El siguiente ejemplo demuestra como ejecutar uno de cuatro subprogramas: PROG0, PROG1, PROG2 o PROG3, dependiendo de que BX contenga el valor 0,1,2 o 3.

```
CMP     BX,4
JNC     ERROR
SHL     BX,1
MOV     BX,TABLA[BX]
JMP     BX
```

```
TABLA    DW    PROG0
         DW    PROG1
         DW    PROG2
         DW    PROG3
```

Un segundo tipo de instrucción de transferencia de control extremadamente importante es la instrucción **CALL**, y actúa de la siguiente manera: supongamos que la CPU encuentra CALL SUBPROG, donde SUBPROG es la dirección incial de una subroutina (es decir, una parte de un programa o subprograma); y queremos que la CPU la ejecute. Esta ejecuta la instrucción guardando primero el contenido actual de IP en un área de memoria llamada Pila. IP contiene la dirección de la instrucción siguiente, es decir, la inmediátamente siguiente a la instruccion CALL. Luego de esto, IP es cargado con la dirección de SUPROG, de modo que la primera instrucción de la subroutina es la siguiente a ejecutarse. La subroutina debe finalizar con una instrucción **RET** (Return). Cuando esta instrucción es ejecutada, la CPU recupera la dirección siguiente de CALL SUBPROG previamente guardada en la pila y la coloca en IP. Por lo tanto, una instrucción CALL simplemente causa que el computador ejecute un subprograma y entonces retorne al programa principal en la instrucción inmediátamente después de la instrucción CALL.

En el Lenguaje Ensamblador, normalmente la instrucción CALL es seguida por un nombre simbólico (una etiqueta), la cual, si es seleccionada cuidadosamente, puede recordar al

programador el propósito de la subroutina. Otra forma de la instrucción CALL en el Lenguaje Ensamblador es CALL mem/reg. El operando mem/reg para la instrucción CALL funciona de la misma manera como fue descrito para la instrucción JMP mem/reg.

Hemos dicho que la CPU almacena el contenido actual del registro IP en la pila cuando una instruccion CALL es ejecutada. Pero, ¿ que es la pila ?. Una Pila es un área de memoria donde se almacena consecutivamente una secuencia de datos. Técnicamente hablando, una pila es una estructura LIFO (Last In, First Out).

Dos operaciones corresponden a la escritura y lectura de datos hacia o desde la pila: **PUSH** y **POP**. PUSH reg16, donde reg16 es un registro de 16 bits, toma el contenido del registro indicado y lo almacena en la localización de memoria dada por el tope de pila. La dirección contenida en el registro puntero de pila SP (Stack Pointer) es la localización del tope de pila. Cuando se ejecuta PUSH AX, este registro SP es automáticamente decrementado en dos, y luego el contenido del registro AX es puesto en la localización de memoria apuntada por SP. Así, SP apunta a un nuevo tope de pila. Puesto que SP es decrementado en dos, el tope de pila está siempre dirigiéndose hacia la localización más baja de la memoria asignada a la pila. El SP es decrementado en dos porque siempre 2 bytes son almacenados por una instrucción PUSH. No es posible hacer PUSH AL. La

figura 3.4 muestra esquemáticamente cómo una operación PUSH afecta a la pila. Además de PUSH reg16, la forma PUSH mem es también permitida; por ejemplo PUSH DATO, donde DATO debe ser el nombre de una variable de 2 bytes.

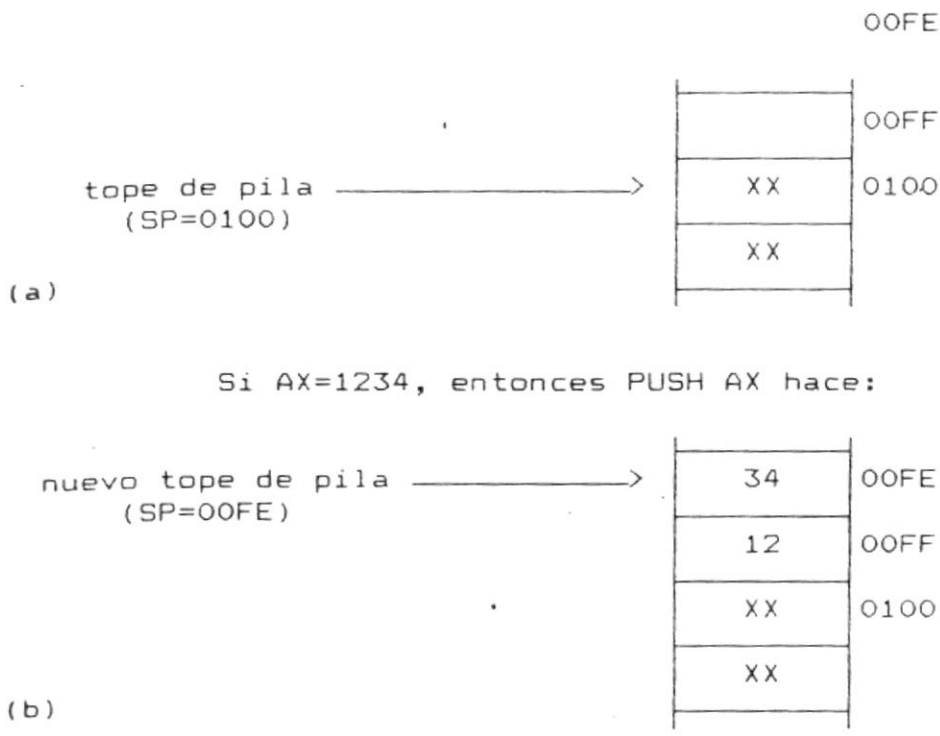


Fig. 3.4 Operación de una pila. (a) La condición inicial de la pila. (b) La pila después de una operación PUSH.

Lo opuesto de la instrucción PUSH es la instrucción POP. La instrucción POP AX lee 2 bytes desde el tope de la pila y los almacena en AX. Aquí, el SP es incrementado en 2 de modo que apunta a un nuevo tope de pila. Al igual que la instrucción PUSH, son aplicables las formas POP mem y POP reg16 donde reg16 es un registro de 16 bits.

La instrucción PUSF guarda en la pila el contenido del registro de estado, mientras que POPF saca 2 bytes del tope

de pila y los deposita en el registro de estado. Estas instrucciones se usan para guardar el condición original de los indicadores de estado antes de ejecutar alguna operación que los pueda modificar y luego recuperarlos a su estado inicial después de finalizada la operación.

La pila es necesaria para realizar llamados a subroutinas. Cuando una instrucción CALL es ejecutada, el contenido de IP es automáticamente puesto en el tope de la pila. Si en la subroutina se alcanza la instrucción RET, los dos bytes del tope de la pila son sacados hacia IP. Si ocurre un segundo CALL en la subroutina antes que una instrucción RET, nuevamente el contenido de IP es almacenado en la pila y el retorno de la primera subroutina no podrá ejecutarse mientras no finalize la segunda con una instrucción RET.

Finalmente podemos decir que la pila es una estructura considerablemente poderosa. Lenguajes de alto nivel hacen bastante uso de la pila y a menudo usan múltiples pilas en memoria.

3.7 INTERRUPCIONES DE PROGRAMA.

Una instrucción muy relacionada con la instrucción CALL es la instrucción **INT**. Básicamente es un tipo especial de llamado indirecto. La ejecución de una instrucción INT nn causa lo siguiente: el registro de las banderas, el registro de segmento CS y el registro IP son almacenados en la pila. IP y CS son entonces cargados con el contenido de los 4 bytes

localizados en las direcciones absolutas de memoria **0:nn*4** y **0:nn*4+2** respectivamente. Así, la ejecución salta a la dirección contenida en esas localizaciones de memoria absolutas. Lo que sucede es que las primeras 400H localizaciones de memoria (0:0 hasta 0:3FFH) se usan para contener las direcciones iniciales de las subroutinas que se deseen ejecutar. Cada grupo de 4 localizaciones contiene una dirección de segmento y un desplazamiento para una subroutina como se muestra en la figura 3.5. Cada grupo es llamado un **vector de interrupción.**

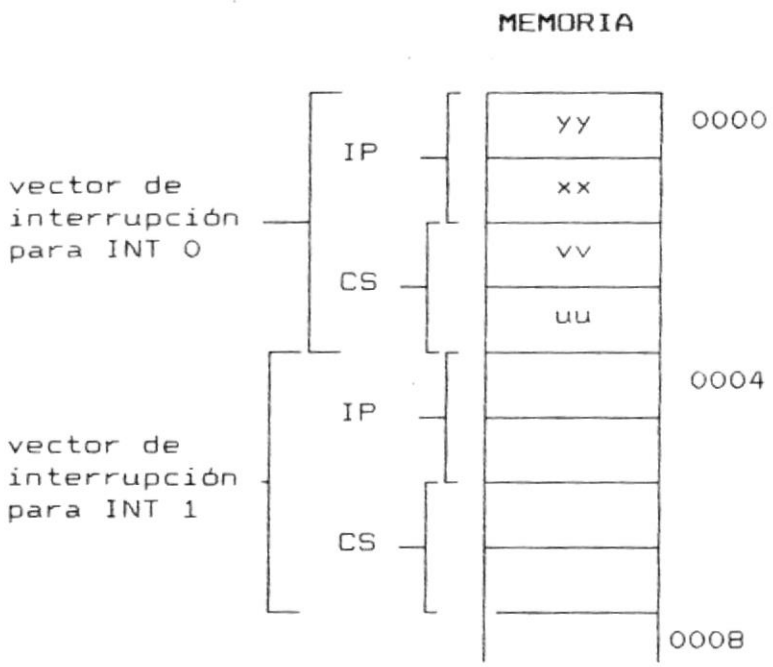


Fig 3.5 Estructura del área del vector de interrupciones en memoria baja. Las direcciones iniciales de INT 0 son CS:IP=xxyy:uuvv.

Los saltos a subroutinas mediante una instrucción INT deben finalizar con una intrucción **IRET** (Return From Interrupt

Instruction). IRET recupera los registros CS, IP y el registro de las banderas.

Otra instrucción de interrupción especial es la interrupción condicional **INTO** (**INT**errupt on **O**verflow), la cual causa una interrupción de programa si la operación previa ha fijado a 1 la bandera de sobrecarga (overflow) .Esta interrupción es útil cuando se usen operaciones aritméticas con signo.

Finalmente, existen dos operaciones que fijan una bandera llamada 'Bandera de Interrupciones' **IF**. Estas son: **CLI** (**CL**ear **I**nterrupts) despeja la bandera de interrupción (la hace 0) y la instrucción **STI** (**S**et **I**nterrupts) que fija la bandera de interrupción (la hace 1). Esta bandera se usa junto con las interrupciones de circuitería. Si la bandera IF es 1, los pedidos de interrupción de dispositivos externos tales como unidades de disco, teclado, etc, serán reconocidos y atendidos. Sin embargo, si IF es 0, las interrupciones de la circuitería serán ignoradas hasta que IF sea fijada otra vez a 1.

3.8 SALTOS CONDICIONALES Y REGISTRO DE ESTADO.

Los saltos y llamados descritos en la sección 3.5 causan que la ejecución del programa salte a otra parte del mismo cada vez que estos son encontrados. Sin embargo, a menudo queremos saltar o no saltar, dependiendo de lo que ha sucedido anteriormente en el programa. El 8088 tiene un conjunto de instrucciones de saltos condicionales. Mediante estas

instrucciones, un salto se realiza o no dependiendo del estado de uno o más bits de indicadores de estado. Por ejemplo, la instrucción JC AQUI causa que la ejecución del programa salte a la localización AQUI solo si el indicador de acarreo es 1. Si este indicador es 0, ninguna acción de salto se realiza y se ejecuta la instrucción inmediátamente siguiente de JC AQUI.

Obviamente, debemos tener una idea clara de como trabajan los indicadores de estado para usar efectivamente los saltos condicionales. Como mencionamos anteriormente, las banderas son indicadores del estado corriente del microprocesadror para un chequeo de las operaciones previas que éste realiza. Por ejemplo, si el microprocesador ejecuta la instrucción ADD AL,5 (sumar 5 al registro AL), algunas banderas serán fijadas a 1 o a 0, dependiendo del resultado de la suma. La bandera del cero (ZF) será 1 si la suma es igual a 0 en AL; de otro modo será 0. La bandera del acarreo será 1 si la suma produce un numero mayor que FF en AL; caso contrario esta será 0. Aquí, durante la ejecución de la instruccion ADD, las banderas son alteradas para reflejar el resultado de la suma.

Las intrucciones que sigan a ADD pueden chequear las banderas y usarlas para hacer decisiones que dependan del resultado de la suma. Si ADD AL,5 es seguida de JZ AQUI, el salto se realizará sólo si ADD AL,5 fue igual a cero.

Existe un total de nueve indicadores diferentes. Cada una es un bit de un registro llamado el Registro de Estado. Este

registro Puede ser almacenado en la pila y recuperado usando las instrucciones **PUSHF** y **POPF**. Este registro no puede ser manipulado igual que los otros registros de propósito general. Tres de los indicadores tienen un propósito muy especial y solo 2 de los 6 restantes son importantes para el 95 por ciento de los programas.

Los tres indicadores de propósito especial son: el indicador de Interrupciones, el indicador de Dirección y el indicador de detención de programa. El indicador de interrupciones **IF** indica si las interrupciones están o no habilitadas. El indicador de dirección **DF** se usa junto con las instrucciones de cadenas discutidas en la sección 3.11. El indicador de detención de programa **TF** permite correr programas de diagnósticos tales como DEBUG. Cuando este indicador es 1, causa que una interrupción ocurra después de la ejecución de cada instrucción de un programa.

Los seis indicadores restantes se usan en operaciones lógicas y aritméticas. Las operaciones aritméticas consisten de las formas usuales de suma, resta, multiplicación y división; incremento, decremento y negación. Las operaciones lógicas consisten de AND, OR, XOR, NOT con algunas instrucciones de desplazamiento junto con la instruccion TEST.

El indicador de Acarrero y el indicador de Cero son los más importantes. Estos dos indicadores pueden usarse cuando se desee controlar lazos y saltos condicionales. Existen 3 instrucciones que manejan el indicador de acarreo: **STC** (Set

Carry Flag) fija este indicador a 1, **CLC** (Clear Carry Flag) lo fija a 0 y **CMC** (Complement Carry Flag) intercambia el indicador de acarreo a 1 si este es 0, o a 0 si es 1.

Pongamos ahora muestra atención a los saltos condicionales en donde realmente se usan las banderas. Observando la tabla 3.5 veremos que existen dos grupos de instrucciones de saltos condicionales, uno para números con signos y otro para números sin signo; además de algunas instrucciones de lazo LOOP.

INSTRUCCIONES DE TRANSFERENCIA DE CONTROL CONDICIONALES

JA	etiqueta	JG	etiqueta
JAE	etiqueta (o JNC etiqueta)	JGE	etiqueta
JE	etiqueta (o JZ etiqueta)	JLE	etiqueta
JNE	etiqueta (o JNZ etiqueta)	JL	etiqueta usadas
JBE	etiqueta		para
JB	etiqueta (o JC etiqueta)	JS	etiqueta números
		JNS	etiqueta con signo
		JO	etiqueta
		JNO	etiqueta

LOOP	etiqueta
LOOPZ	etiqueta (o LOOPE etiqueta)
LOOPNZ	etiqueta (o LOOPNE etiqueta)

JCXZ	etiqueta
JPO	etiqueta (o JNP etiqueta)
JPE	etiqueta (o JP etiqueta)

donde 'etiqueta' es una etiqueta de programa

INSTRUCCIONES DE LA BANDERA DE ACARREO

CLC
STC
CMC

Tabla 3.5 Instrucciones de transferencia de control y bandera de acarreo.

El primer grupo de saltos condicionales se usa para números binarios sin signo (solo positivos). Estas instrucciones tienen los siguientes significados.

JA (Jump if Above)

JAE (Jump if Above or Equal) o JNC (Jump if No Carry)

JE (Jump if Equal) o JZ (Jump if Zero)

JNE (Jump if Not Equal) o JNZ (Jump if Not Zero)

JBE (Jump if Below or Equal)

JB (Jump if Below) o JC (Jump if Carry)

Es preferible usar las formas JC, JNC y JNZ junto con alguna instrucción lógica o aritmética en lugar que con una instrucción de comparación. Por ejemplo, la siguiente secuencia de instrucciones suma los enteros 1 hasta 9.

```
        MOV     BL,9
        MOV     AL,0
SIGUIENTE:
        ADD     AL,BL
        DEC     BL
        JNZ     SIGUIENTE
```

La instrucción **JNZ** crea un lazo que se repite hasta que BL sea decrementado hasta cero. Existe una instrucción que realiza esta operación en la cual una cantidad es decrementada hasta cero en un lazo. Esta es la instrucción **LOOP** la cual actúa como una combinacion de las instrucciones DEC y JNZ. Usando la instrucción LOOP, el programa anterior

será:

```
        MOV     CX,9
        MOV     AX,0
SIGUIENTE:
        ADD     AX,CX
        LOOP    SIGUIENTE
```

Se ha usado el registro CX porque la instrucción LOOP trabaja solamente con este registro. Lo interesante de esta instrucción es que no afecta a las banderas. LOOP "ve" diréctamente si el registro CX es cero. Esto puede ser útil en lazos en donde las banderas son usadas para otras tareas.

Existen dos variaciones de menor orden que la instrucción LOOP: **LOOPZ** (Loop if Zero) y **LOOPNZ** (Loop if not Zero). LOOPZ decrementa CX y salta a la dirección del operando si CX es diferente de cero y si la bandera del cero es 1. Por otro lado, LOOPNZ decrementa CX y salta a la dirección del operando si CX es diferente de cero y si la bandera del cero es igual a 0.

Una instrucción usada con la instrucción LOOP es la instrucción **JCXZ** (Jump if **CX** Zero). JCXZ ejecuta un salto solo si CX es cero. Esta instrucción es de mucho valor en una situación muy común. Supongamos que queremos repetir una acción cierto número de veces usando la instruccion LOOP y el valor inicial en el registro CX es un número que depende del resultado de un cálculo previo. Cuando el valor en CX es diferente de cero, la acción se repite CX veces; pero si

sucede lo contrario, la instrucción LOOP decrementa CX desde cero hasta FFFF H, y el lazo se realizará 65.356 veces..! Este problema puede ser eliminado usando una instrucción JCXZ al inicio del lazo.

```
            ....            ;alguna instrucción que
                            ;inicializa CX
            JCXZ    CONTINUE
SIGUIENTE:  ....            ;el lazo comienza aqui
            LOOP    SIGUIENTE
CONTINUE:   ....            ;el programa continúa aqui.
```

Una limitación en todas las instrucciones de salto condicional es que el rango del salto está limitado a 127 bytes hacia atrás del programa o 128 bytes hacia adelante. Si se trata de usar un salto a una localización mayor que este rango, el Ensamblador da un mensaje de error.

Ahora veamos el segundo set de saltos mostrado en la tabla 3.5. Estos saltos condicionales solo se usan con números binarios con signo. Sus significados son:

JG (Jump if Greater)

JGE (Jump if Greater or Equal)

JLE (Jump if Less or Equal)

JL (Jump if Less)

JS (Jump if Sign)

JNS (Jump if Not Sign)

JO (Jump if Overflow)

JNO (Jump if Not Overflow)

Las instrucciones JS y JNS se usan para ejecutar un salto solo si la bandera del signo es 1 o 0. Se pueden usar estas instrucciones para ejecutar un salto cuando un número es positivo o negativo.

Por último, existen instrucciones de salto muy poco utilizadas: JPO (Jump if Parity Odd) y JPE (Jump if Parity Even). Estas instrucciones ejecutan el salto dependiendo del estado de la bandera de paridad PF. Esta bandera es 1 si el resultado de una operación tiene un número par de "unos". Si existe un número impar de 1's en el resultado, PF es fijada a cero. Estas instrucciones se pueden usar para controlar un puerto de comunicación serial. Cuando un byte de datos es enviado al puerto serial, a veces un bit de paridad es enviado ademas del byte. Este bit de paridad trabaja igual que la bandera de paridad. Este es fijado a 1 si el byte contiene un número par de 1's. El programa que usa el puerto serial puede chequear este bit para ver si el byte ha sido recibido corréctamente y saltar a una rutina de error si no es asi.

3.9 INSTRUCCIONES ARITMETICAS

El 8088 tiene un buen grupo de instrucciones aritméticas las cuales son resumidas en la tabla 3.6. Las instrucciones más básicas son las operaciones ADD (suma) y SUB (resta). Estas instrucciones soportan casi todos los modos de direccionamiento. ADD y SUB pueden usarse igualmente con cantidades de 8 y 16 bits. Por ejemplo, ADD BL,CH suma CH a

BL colocando el resultado en BL, mientras que SUB SI,1234 resta el valor 1234 de SI, dando el resultado en SI. El destino para el resultado puede ser un registro o una localización de memoria.

INSTRUCCIONES ARITMETICAS

ADD	reg,reg/mem	ADC	reg,reg/mem	
ADD	mem.reg	ADC	mem,reg	
ADD	reg/mem,reg	ADC	reg/mem,reg	
SUB	reg,reg/mem	SBB	reg,reg/mem	
SUB	mem,reg	SBB	mem,reg	
SUB	reg/mem,mem	SBB	reg/mem,mem	
CMP	reg,reg/mem			
CMP	mem,reg			
CMP	reg/mem,reg			
INC	reg/mem	DEC	reg/mem	
NEG	reg/mem			
MUL	reg/mem	IMUL	reg/mem	
DIV	reg/mem	IDIV	reg/mem	usadas para operaciones con signo
		CBW		
		CWD		

Tabla 3.6 Instrucciones aritméticas.

Puesto que la longitud de los operandos puede ser de 8 o 16 bits, el resultado de una operación aritmética puede exceder la capacidad del registro o localidad de memoria. Por ejemplo, si AX contiene el valor 0F910H y se ejecuta la instrucción ADD AX,800H, el resultado, 10110H, es demasiado grande como para caber en 16 bits. En este caso la CPU fija la bandera de acarreo a 1, y únicamente los 16 bits menos significativos (0110H) son colocados en AX. Si los números

son interpretados como números binarios con signos, se debe tener más cuidado puesto que la suma de dos números positivos de 16 bits que exceda 7FFFH dará un resultado incorrecto. Asi mismo se debe tener cuidado con la operación de resta. Si AL contiene 20H y BL contiene 36H, la operación SUB AL,BL fija la bandera de acarreo para indicar que un préstamo en el bit de más alto orden ha ocurrido, y el número EAH (el cual es -16H en la forma de complemento a 2) es colocado en AL. Este resultado es correcto si los números son interpretados como números binarios con signo, pero no en caso contrario. En algunos casos, la instrucción SUB parece redundante puesto que un número puede restarse de otro sumando éste como un número de complemento a 2. En efecto, existe una operación aritmética que produce el complemento a 2 de un número. Esta es llamada **NEG** reg/mem, y forma el negativo de un número binario de 8 o 16 bits. Asi, NEG BX seguido de ADD AX,BX produce el mismo resultado que SUB AX,BX. Sin embargo, el uso de SUB es más rápido, asi que NEG se usa solo para cambiar el signo de un número. Las operaciones **ADC** (**Add** with **C**arry) y **SBC** (Subtract with **B**orrow) se usan para facilitar operaciones de múltiples bytes. La instrucción ADC ejecuta la suma de dos operandos y la bandera de acarreo. Por otro lado, SBB OP1,OP2 realiza la operación OP1-OP2-CY.

Una instrucción relacionada con S B es la instrucción de comparación **CMP**. Esta instrucción resta el origen del destino, fija las banderas apropiadamente para indicar un prestamo, sobrecarga, etc sin que el resultado afecte a los

operandos; únicamente a las banderas. Para ver esto, CMP OP1,OP2 compara OP2 con OP1 y por medio de las banderas indica si OP1 es más grande (CY=1, Z=0), igual (CY=0, Z=1) o menor (CY=0, Z=0) que OP2.

Otras dos instrucciones aritméticas muy comunes son la de incremento **INC** y la de decremento **DEC**. Estas instrucciones suman o restan 1 al contenido de algún registro o localización de memoria.

El segundo conjunto de instrucciones aritméticas del 8088 lo conforman las instrucciones de multiplicación **MUL** e **IMUL** y las instrucciones de división **DIV** e **IDIV**. Estas instrucciones dan la facilidad de multiplicar y dividir números binarios con o sin signo de 8 y 16 bits. MUL realiza la multiplicación de dos números sin signos de 8 bits o dos números sin signos de 16 bits. En el caso de 8 bits uno de los números es un registro de 8 bits o byte de memoria, mientras que el otro número debe estar contenido en AL. Aqui, la instrucción solo tiene un operando, MUL OP1. El otro operando (AL) es implicito, y el resultado de la operación es obtenido en AX. De igual forma, si un operando es un registro de 16 bits o una palabra en memoria, el otro operando se asume en AX, y el resultado de la multiplicación es colocado en DX (16 MSB) y AX (16 LSB). La figura 3-6 muestra esquemáticamente los registros usados para las instrucciones de multiplicación de 8 y 16 bits. IMUL trabaja de la misma manera que MUL, pero trata los operandos como números binarios con signos.

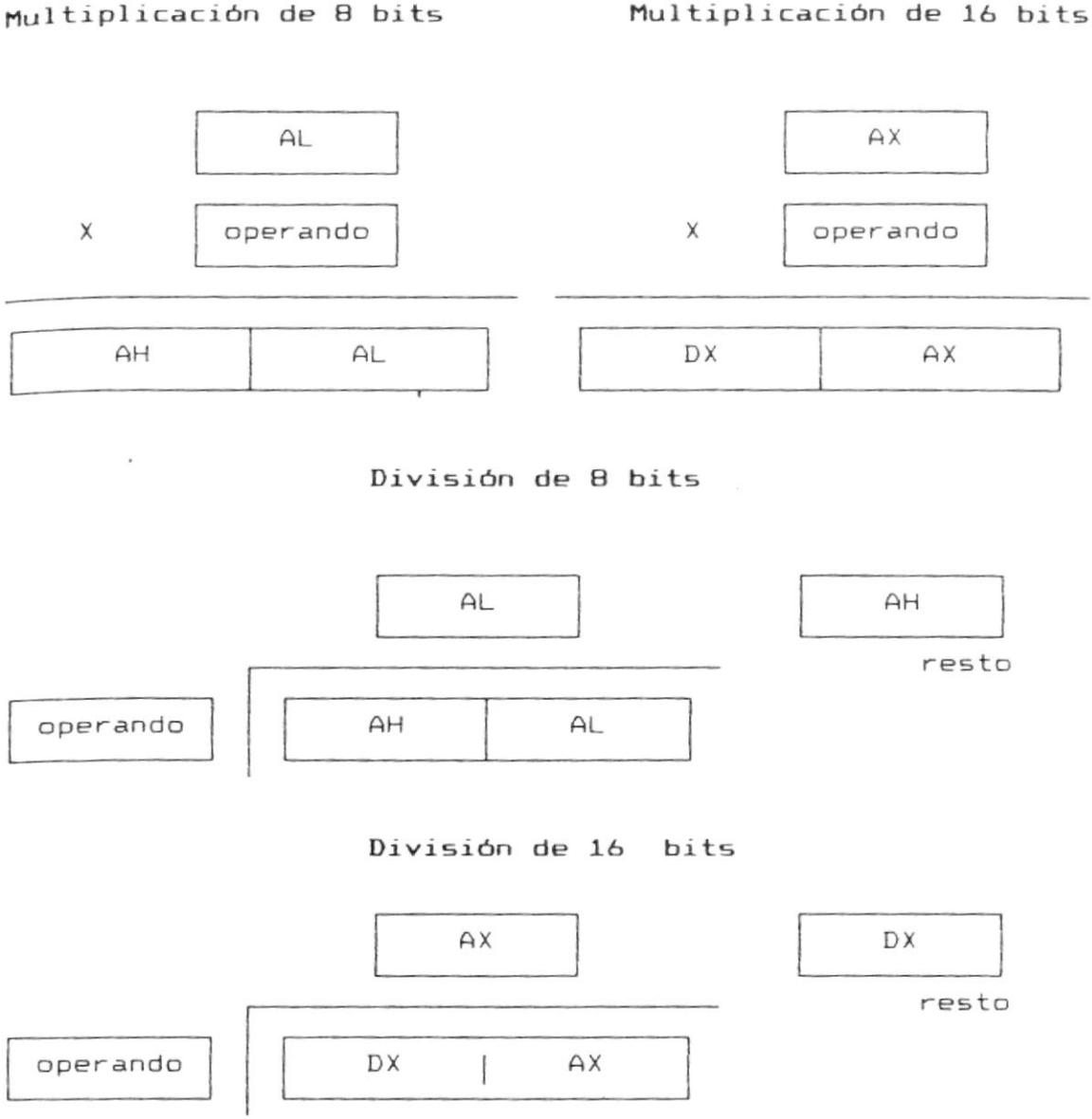


Fig. 3.6 Registros usados en instrucciones de multiplicación y división.

DIV e **IDIV** operan de una manera similar a MUL e IMUL. El dividendo se asume en AX para dividirse por un valor en un registro de 8 bits o un byte de memoria. Cuando la división es de un valor de 16 bits el dividendo se asume en un valor de 32 bits contenido en DX (MSB) y AX (LSB). El cociente es almacenado como un entero en AL (AX para división de 16

bits), y el resto en AH (DX para división de 16 bits).

Finalmente consideremos dos instrucciones que se usan con la instrucción IDIV ,llamadas **CBW** (Convert Byte to Word) y **CWD** (Convert Word to Double-word). CWD examina el byte contenido en AL y, dependiendo del bit más significativo (bit del signo), fija AH a 00 o FF. CWD opera de la misma manera excepto que ésta convierte dos bytes en AX a una palabra de doble longitud en los registros DX:AX. El registro DX es fijado a 0000 o 0FFFFH, dependiendo del bit del signo del valor en AX. Esta instrucción es necesaria cuando se desea dividir un valor con signo en AX por un número de 16 bits.

3.10 INSTRUCCIONES LOGICAS

El 8088 posee también un número de instrucciones lógicas que permiten realizar operaciones de programa iguales a las operaciones lógicas realizadas por las puertas lógicas de la circuitería. La tabla 3.7 resume esta clase de instrucciones.

INSTRUCCIONES LOGICAS

AND	reg,reg/mem	TEST	reg,reg/mem
AND	mem,reg	TEST	mem,reg
AND	reg/mem,mem	TEST	reg/mem,mem
OR	reg,reg/mem		
OR	mem,reg		
OR	reg/mem,mem		
XOR	reg,reg/mem		

INSTRUCCIONES LOGICAS			
SHR	reg/mem,1	SHL	reg/mem,1
SHR	reg/mem,CL	SHL	reg/mem,CL
SAR	reg/mem,1	SAL	reg/mem,1
SAR	reg/mem,CL	SAL	reg/mem,CL
ROR	reg/mem,1	ROL	reg/mem,1
ROR	reg/mem,CL	ROL	reg/mem,CL
RCR	reg/mem,1	RCL	reg/mem,1
RCR	reg/mem,CL	RCL	reg/mem,CL

Tabla 3.7 Operaciones lógicas para el 8088.

Consideremos primero las 3 instrucciones **AND, OR** y **XOR** las cuales son analogías directas de las puertas elementales de dos entradas. La operación AND de dos digitos es un binario 1 solo si el primer bit es 1 y el segundo bit es 1; asi, 1 AND 1=1, 1 AND 0 = 0, y 0 AND 0 = 0. El resultado de la operación OR de dos bits es 1 si uno o ambos bits es igual a 1, asi 1 OR 1 = 1, 1 OR 0 = 1 y 0 OR 0 = 0. La operación lógica XOR (eXclusive OR), de dos bits es 0 si ambos bits son iguales y es 1 si son diferentes; asi, 1 XOR 1 = 0, 0 XOR 0 = 0 y 1 XOR 0 = 1.

Otra instrucción muy útil para operar con bits individuales en un registro o localización de memoria es la instrucción **TEST**. TEST realiza una operación AND de los operandos, y fija las banderas de acuerdo al resultado de la operación sin afectar a los operandos. TEST actúa como una especie de comparación (CMP) que compara bits individuales en lugar de

números. A menudo, TEST se usa para ver si un bit en un registro o localización de memoria es 1 o 0. Por ejemplo una forma de chequear si un número binario con signo de 8 bits contenido en el registro DH es positvo o negativo será ejecutando la instrucción TEST DH,80H. Si el número en DH es positivo, (bit 7 igual a 0), entonces TEST DH,80H fijará la bandera del cero a 1, pero no afectará a DH.

El resto de las instrucciones lógicas lo conforman las instrucciones de desplazamiento y rotación. Estas instrucciones desplazan todos los bits de un registro o localización de memoria hacia la derecha o hacia la izquierda. Las acciones realizadas por cada instrucción está mostrada en la figura 3_7. Veamos primero las dos instrucciones de desplazamiento SHL (SHift logical Left) y SHR (SHift logical Right). SHL AX,1 desplaza cada bit en AX una posición hacia la izquierda. El bit más significativo de AX es desplazado hacia el bit del acarreo, y un cero es ingresado en la posición del bit menos significativo. Esta acción es equivalente a multiplicar el número de AX por 2. Una variación de esta instrucción permite desplazar algunos bit a la vez con las siguientes instrucciones.

```
MOV CL,n        ;Coloca en CL el número de posiciones a
                ;desplazar
SHL AX,CL       ;Desplaza AX a la izquierda n bits
```

Esto es equivalente a ejecutar la instrucción SHL AX,1 n veces. El contenido de CL no es afectado por la instrucción

SHL AX,CL.

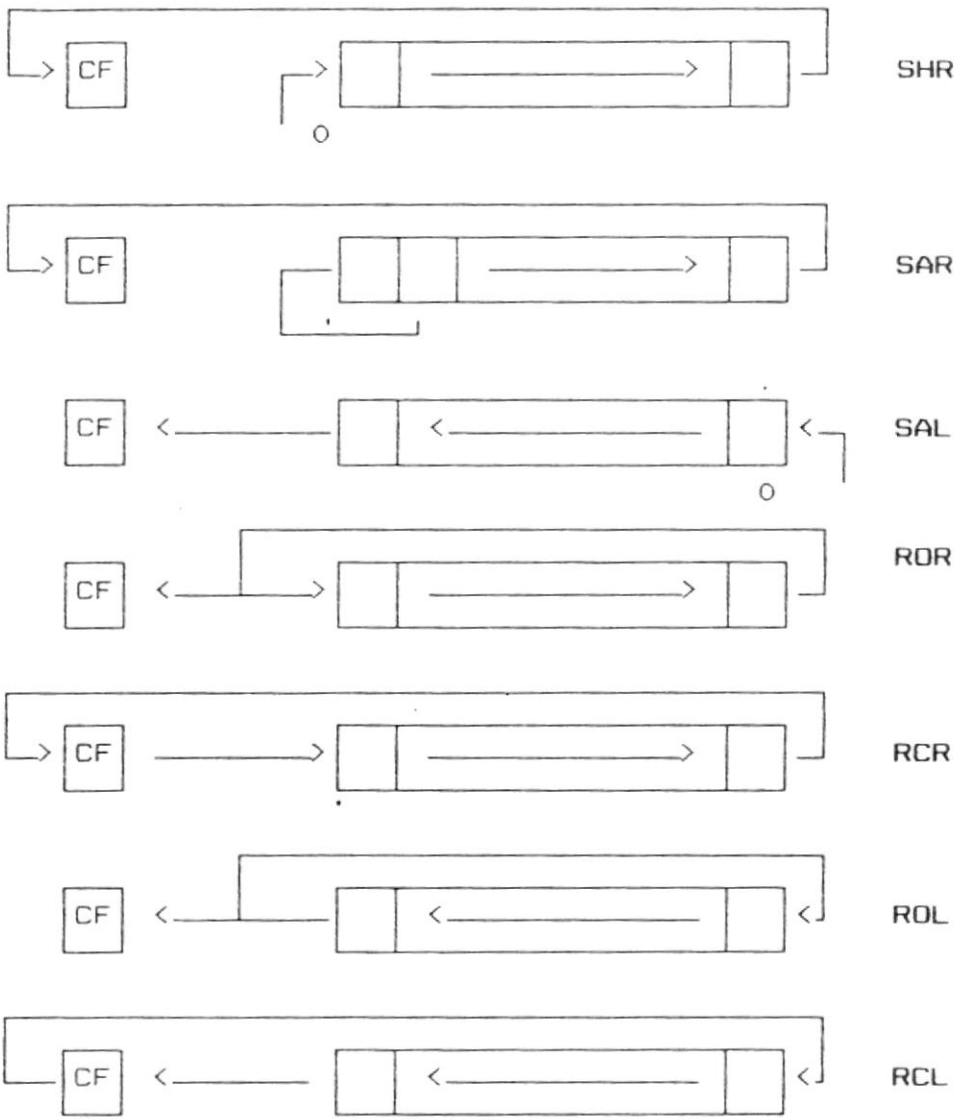


Fig. 3.7 Operaciones realizadas por las instrucciones de desplazamiento y de rotación.

La instrucción SHR opera de una manera similar a SHL solo que todos los bits son desplazados hacia la derecha, el bit menos significativo hacia el indicador de acarreo y un 0 es ingresado en la posición del bit más significativo. Esta instrucción es equivalente a dividir para 2 un número sin

signo.

Otro par de instrucciones de desplazamiento es **SAR** (Shift Arithmetic Right) y **SAL** (Shift Arithmetic Left). La instrucción SAR trabaja igual que SHR con la diferencia de que en vez de poner un 0 en el bit más significativo, la instrucción SAR ignora este bit y su valor no es afectado por la instrucción. Esta instrucción permite dividir para 2 números binarios con signo (un número binario negativo siempre tiene un 1 en el bit más significativo). La instrucción SAL está incluida solo por simetría. Esta realiza una multiplicación por 2 a los números binarios con signo, pero una multiplicación es la misma para números con signo que para números binarios sin signos. Aqui,la instrucción SAL es idéntica a la instrucción SHL y ambas tienen el mismo código de operación de lenguaje de máquina.

Las instrucciones **ROR** (ROtate Right) y **ROL** (ROtate Left) mueven bits de un modo circular. ROR AX,1 desplaza una posición a la derecha los bits en AX y el bit menos significativo es movido hacia la posición del bit más significativo. Ademas, el LSB es copiado en la bandera del acarreo. Ejecutando 4 instrucciones ROR AL,1 se intercambian los 4 bits MBS con los 4 LBS de un byte. La instrucción **ROL** es lo inverso de ROR. ROL rota los bits a la izquierda.

Las dos últimas instrucciones lógicas son **RCR** (Rotate through Carry Right) y **RCL** (Rotate through Carry Left). Estas instrucciones operan igual que ROL con la diferencia de el

indicador de acarreo actúa como un bit extra unido al bit más significativo del byte usado en la instrucción.

3.11 INSTRUCIONES DE CADENAS

Uno de los grupos de instrucciones mas poderosos es el grupo de cadenas primitivas. Estas instrucciones, resumidas en la tabla 3.8 permiten mover bloques de datos rápidamente de un lugar a otro o comparar los bloques de datos.

INSTRUCCIONES DE CADENAS PRIMITIVAS

MOVSB	CMPSB
MOVSW	CMPSW
LODSB	SCASB
LODSW	SCASW

Prefijos para Cadenas Primitivas.

REP
REPE (o REPZ)
REPNE (O REPNZ)

Instrucciones de Bandera de Dirección

CLD
STD

Tabla 3.8 Instrucciones de cadenas primitivas

Se denominan operaciones de cadenas primitivas porque ellas proveen las instrucciones básicas (o primitivas) que pueden combinarse para realizar operaciones complicadas sobre cadenas de datos. Una cadena significa un bloque de datos contenido en direcciones de memoria consecutivas. Consideremos primero la característica de movimiento de datos.

Las instrucciones de movimiento de cadenas (**MOVSB** para mover un byte y **MOVSW** para mover una palabra) mueven datos desde un lugar de memoria a otro. Estas instrucciones asumen que el registro SI está apuntado al origen (el lugar desde donde se quiere mover), y que el registro DI está apuntando al destino (el lugar hacia donde se quiere mover). Por ejemplo, supongamos que se desea mover dos bytes de datos desde una localización de memoria llamada ORIGEN hacia otra localización llamada DESTINO. Es posible realizar esto usando el siguiente código:

```
MOV     SI,OFFSET ORIGEN
MOV     DI,OFFSET DESTINO
MOVSW
```

Nótese que se debe cargar SI Y DI con OFFSET ORIGEN y OFFSET DESTINO (recordemos que OFFSET significa "dirección de") porque ambos registros deben apuntar a las direcciones de ORIGEN y DESTINO respectivamente. Se puede usar la instrucción **LEA** en lugar de la instrucción MOV para cargar las direcciones de origen y destino. Realmente, lo que la instrucción MOVS (MOVSB o MOVSW) realiza es la operación **MOVS [DI],[SI]**. SI es el registro índice origen y DI es el registro índice destino. Asi, el Ensamblador no necesita operando alguno en la instrucción MOVS; solo necesita saber si el movimiento es de uno o de dos bytes. Por eso los sufijos "B" y "W".

Otra cosa importante de la intrucción MOVS, y de todas las

instrucciones de cadenas primitivas, es que el origen y el destino pueden estar en diferentes segmentos. La dirección origen para todas las cadenas primitivas es siempre **DS:SI**, y la dirección del destino es siempre **ES:DI**. A menudo los registros de segmento ES y DS están apuntando a la misma dirección segmento del inicio de un programa, de modo que los registros de segmento pueden ignorarse.

El poder de la instrucción MOVS es más que un simple movimiento de datos. Si el indicador de dirección DF tiene un valor de 0, automáticamente se incrementan los registros SI y DI (1 para MOVBS y 2 para MOVSW), de este modo, SI y DI quedan apuntando al siguiente byte o palabra de memoria. La idea es que MOVS se usa cuando se desea mover más de un byte. Además, la instrucción MOVS puede usarse con un prefijo llamado **REP**, el cual hace que la instrucción de movimiento se ejecute **n** veces, donde n es el valor en el registro CX. Por ejemplo, el siguiente grupo de instrucciones:

```
CLD                     ;Fija DF=0
LEA   SI,ORIGEN         ;apunta a origen
LEA   DI,DESTINO        ;y destino de datos
MOV   CX,20H            ;32 bytes a mover
REP   MOVSB             ;ejecuta el movimiento de los datos
```

Mueve los 32 bytes de datos que comienzan en la localización ORIGEN hacia los 32 bytes a partir de la localización DESTINO. Esta es una manera rápida de mover bloques de datos.

El prefijo REP hace que CX sea decrementado por cada

ejecución de la instrucción MOVSB. La instrucción REP MOVSB es equivalente al siguiente grupo de instrucciones:

```
LAZO: MOV    AL,[SI]
      MOV    [DI],AL
      INC    SI
      INC    DI
      LOOP   LAZO
```

y la diferencia está en que REP MOVSB se ejecuta mucho más rápido y requiere menos bytes de código de máquina.

Como se indicó anteriormente, la instrucción MOVS incrementa automaticámente SI y DI si la bandera DF tiene un valor de cero. Pero, si DF tiene un valor 1, los registros SI y DI serán decrementados 1 o 2 posiciones según la instrucción que se esté usando (MOVSB o MOVSW).

Existen dos instrucciones que afectan a la bandera de dirección DF, denominadas CLD (Clear DF) y STD (Set DF). STD fija DF a 1 y CLD la fija a 0.

Otra instrucción de cadena es la instrucción de almacenamiento en cadena **STOS**. SOTOSW mueve el valor contenido en AX hacia la palabra en memoria direccionada por el registro DI e incrementa este registro (o lo decrementa si DF=1) dos posiciones. Por otro lado, la instrucción STOSB opera de igual forma, pero solc usa el registro AL para almacenamiento de un byte en memoria.

La instrucción de lectura de cadenas **LODS** es lo contrario de

STOS. LODSW mueve la palabra de memoria direccionada por SI hacia el registro AX e incrementa SI (o lo decrementa si DF=1) dos posiciones. LODSB en cambio mueve el byte contenido en la localización de memoria direccionada por SI hacia el registro AL.

Las dos últimas instrucciones de cadena primitiva son las instrucciones de comparación de cadenas **CMPS**, y la instrucción de observación de cadenas **SCAS**. La instrucción CMPSB compara el byte apuntado por SI con el byte apuntado por DI y fija las banderas para indicar que el contenido de [SI] es mayor que, igual que, o menor que el contenido de [DI]. Los dos bytes no son afectados por esta instrucción, pero sí los registros SI y DI que son automáticamente incrementados o decrementados de acuerdo al valor del indicador de dirección DF.

La comparación se realiza restando el byte apuntado por DI del byte apuntado por SI, es decir, calculando [SI] – [DI], con lo cual los indicadores reflejarán el estado de la operación. De igual forma funciona la instrucción CMPSW, pero con 2 bytes.

La otra instrucción de comparación es SCAS. SCASB compara el byte en memoria direccionada por DI con el byte en AL, fija las banderas e incrementa o decrementa DI. Mientras CMPSB compara los bytes correspondientes de dos cadenas en memoria, SCASB compara los byte de una cadena con un valor fijo en AL.

La comparación se realiza restando el byte de memoria del

byte en AL, es decir, AL←[DI]. Solamente los indicadores de estado son afectados por esta intrucción. SCASW ejecuta la misma operación, excepto que la comparación es entre una palabra en memoria con el contenido del registro AX.

Con las instrucciones CMPS y SCAS pueden usarse los prefijos REPE y REPNE (repetir mientras sea igual a/o no sea igual a). El IBM Macro Ensamblador también reconoce REPZ y REPNZ (repetir mientras sea igual a cero/o mientras no sea igual a cero) como sinónimo de REPE y REPNE respectivamente. Estos prefijos hacen que la instrucción CMPS se repita n veces, donde n es el número de veces contenido en el registro CX. Este registro es decrementado cada vez que se ejecuta la instrucción CMPS.

El mayor grupo de instrucciones del 8088 ha sido discutido en las secciones previas. Existen muchas otras intrucciones por discutir, tales como las de ajuste decimal y las que involucran a los registro de segmento; sin embargo consideramos no necesario detallar cada una de ellas puesto que son muy poco utilizadas

CAPITULO 4

COMUNICACION DEL IBM PC CON LA LINEA DE TELEX

4.1 INTRODUCCION

Una vez asimilada la terminología básica de comunicaciones podemos emprender una explicación detallada de las especificaciones de uso del IBM PC en entornos de comunicación asincrónica. En este capítulo se proporciona una idea completa de los dispositivos necesarios para llevar a cabo con éxito una comunicación asincrónica con el computador personal.

IBM ofrece un adaptador, conocido como adaptador de comunicación asincrónica, que incluye una interfase RS-232C. Este adaptador proporciona velocidades de transmisión comprendidas entre los 50 y 115200 bauds. Las direcciones 3F8h y 2F8h, pertenecientes a los puertos COM1 y COM2 se emplean para accesar al puerto del adaptador. Este adaptador está equipado con un conector DB25P o DB9P macho en la placa, que sobresale por detrás del PX/XT/AT (en los IBM PS/2 y algunos comutadores compatibles viene incluido en la tarje'a principal del sistema).

En entornos asincrónicos, todos los datos que salgan del

computador atravezarán una vía llamada línea de datos transmitidos. Hay otra línea conocida como línea de datos recibidos, reservada para todos los datos que ingresan. En una interfase RS-232C hay 25 pines o líneas disponibles. Sin embargo, solo se usan algunas de estas líneas como "vías" para la transferencia de datos. Cada línea tiene asignada una función (tal como se, describió en el capitulo 1). Por ejemplo, los datos transmitidos van por la línea 2, y los datos recibidos por la línea 3.

A continuación se proporciona una descripción de la forma como se realiza el envio y recepción de información a través de un puerto serial, utilizando el UART 8250.

4.2 CONVERSION PARALELO/SERIE/PARALELO: EL UART 8250

La parte principal de un puerto serial es un integrado denominado el Transmisor Receptor Asincrónico Universal, comunmente conocido como **UART**. Este es el UART que físicamente transmite y recibe caracteres a través de los pines de un conector RS232-C. Cuando se intenta transmitir un caracter, el UART agrega a este caracter ciertos bits previamente definidos tales como: bit de inicio, bit de parada y bit de paridad; y luego coloca este caracter bit a bit en un punto de transmisión de datos a intervalos predefinidos.

De la misma forma, cuando un dato de entrada es recibido en el pin de recepción de datos, el UART descompone el paquete y lo presenta en la forma de un caracter de 5, 6, 7 u 8 bits.

El UART también suministra un número de funciones de alto nivel que ayudan a colocar caracteres de entrada/salida bajo el control de programas. Si es necesario, éste puede comparar el bit de paridad de cada caracter recibido, con otro valor de paridad que calcule, reportando un error de paridad si no son iguales. El UART también puede reportar un error de sobrecarga de recepción si un caracter recibido no es tomado desde el espacio interno del UART antes de que el siguiente caracter sea detectado, o un error de formato de recepción del caracter en el caso de recibir un bit de parada incorrecto.

Los registros internos del UART conectados a los pines de control RS232 permiten a los programas realizar control de comunicación con dispositivos conectados al puerto serial. Por ejemplo, un dispositivo transmitiendo puede activar su pin RTS (Request To Send), indicando a otro dispositivo que está listo para enviar un caracter por la linea de comunicación a la que están conectados. Antes de enviar el caracter, el dispositivo transmisor puede requerir la confirmación de que el dispositivo de recepción está listo para recibirlo. El UART suministra esta señal de confirmación como CTS (Clear To Send), controlada por el dispositivo receptor.

El UART permite ser programado para generar una interrupción cuando ocurra un evento de entrada/salida. Mientras tanto, el un controlador de interrupciones denominado el 8259 es programado para pasar la interrupción hacia la CPU fijando

bits selecionados en su Registro de Habilitación de Interrupciones. Se puede programar el UART para que genere una interrupción bajo una variedad de condiciones: Cuando su Registro de Recepción de Datos reciba un caracter, cuando el Registro de Transmisión esté listo para recibir otro caracter desde la CPU, cuando ocurra un error de paridad, error sobrecarga o mal formato del caracter, o cuando cambie el estado de uno de los pines de control del puerto RS232.

Los programas de aplicación fijan sus propios manipuladores de interrupción para atender a cada interrupción que estos seleccionen. Si más de un tipo de interrupción es habilitado, el programa puede determinar qué condición fue la que ocurrió más reciéntemente, leyendo un registro del UART llamado Registro de Identificación de interrupciones.

La tabla 4.1 muestra los registros que internamente posee el UART. Dentro del computador personal, a partir de la dirección 0:400, existe una tabla que contiene las direcciones de 16 bits de entrada/salida de todos los puertos seriales instalados en el sistema. Una vez obtenida la dirección base del UART, sus registros internos pueden ser accesados con relación a la base. Por ejemplo, si el computador tiene instalados los puertos seriales COM1 y COM2, usando el programa DEBUG suministrado por el DOS podemos observar que a partir de la dirección 0:400 estan almacenadas las direcciones relativas de cada puerto; esto es: 03F8h y 02F8h respectivamente. Un valor 0000 en estas direcciones indica que el correspondiente puerto serial no está

instalado.

LOS REGISTROS DEL UART	
Dirección Base	
+ 0	Registro de Transmisión/Recepción de caracteres
+ 1	Registro de Habilitación de Interrupciones
+ 2	Registro de Identificación de Interrupciones
+ 3	Registro de Control de Linea
+ 4	Registro de Control de Modem
+ 5	Registro de Estado de Linea
+ 6	Registro de Estado de Modem
+ 0 *	Registro Divisor LSB de Baud Rate
+ 1 *	Registro Divisor MSB de Baud Rate

Tabla 4.1: El asterisco indica que los registros son mapeados a esa dirección cuando el bit DLAB sea igual a 1.

La tabla 4.2 muestra la forma como el UART genera e identifica las interrupciones. Además del Registro de Habilitación de Interrupciones y el Registro de Identificación de Interrupciones, los registros de Recepción de Datos, de Transmisión de datos, de Control de Linea y Control de Modem también juegan 1 papel importante en el manejo de las interrupciones en la comunicación serial.

EL Registro de Recepción y Transmisión realiza una doble tarea: un caracter pasado a éste con una instrucción OUT es

transferido al registro de transmisión del UART y enviado; mientras que, un caracter tomado desde éste con una instrucción **IN** es tomado desde el área del registro de Recepción del UART. El Registro de Control de Línea contiene la información del formato del dato enviado o recibido, el cual define el número de bits de datos, el número de bits de parada y la paridad. Además, el bit 7 de este registro sirve como un bit divisor de acceso (Divisor Latch Access Bit) o DLAB.

Cuando DLAB es 0, los puertos relativos 0 y 1 de la dirección base corresponden a los registros de recepción de transmisión y de habilitación de interrupciones respectivamente; cuando DLAB es 1, las mismas direcciones mapean a los registros que contienen la velocidad de transmisión. Una rutina de servicio de interrupción que lea el puerto serial, debe por lo tanto asegurarse que DLAB sea igual a cero antes de leer un caracter con una instrucción IN desde la dirección relativa 0.

Finalmente, el Registro de Control de Modem contiene los bits de control DTR (Data Terminal Ready) y RTS (Request to Send). Este también contiene el bit de salida de propósito general GP02 en la implementación IBM. Este bit debe ser igual a 1 para que las interrupciones del UART sean reconocidas por la CPU. El Registro de Estado de Modem contiene los pines de entrada RS232, DSR (Data Set Ready), CTS (Clear To Send), RI (Ring Indicator) y DCD (Data Carrier Detect).

Para enviar o recibir datos apropiadamente desde el puerto de datos del UART 8250 (dirección 03F8h para COM1 y 02F8h para COM2), es necesario conocer como activar las líneas de control de salida DTR y RTS, como chequear las líneas de control de entrada DSR y CTS, como detectar cuando un dato serial ha sido recibido o cuando los datos pueden ser transmitidos.

El control de las líneas DTR y RTS se realiza a través del registro de Control de Modem (COM1=03FCh, COM2=02FCh). Para fijar la salida DTR a nivel bajo, indicando que el 8250 está operacional, es necesario fijar en 1 el bit 0 de este registro. Similarmente, el bit 1 debe tener un valor de 1 para forzar la salida RTS en un nivel bajo, indicando que el 8250 está listo a recibir datos. Las siguientes instrucciones permiten fijar las señales DTR y RTS al nivel bajo correspondiente.

```
MOV     DX,03FCh        ;Apunta al registro del
MOV     AL,3            ;control de modem
OUT     DX,AL           ;Fija DTR y RTS a nivel bajo.
```

El Registro de Estado de la Línea (COM1=03FDh, COM2=02FDh) se usa para sensar el estado de la línea serial. El bit DR, bit 0, indica si algún dato ha sido recibido, y el bit THRE, bit 5, indica si el registro de transmisión está vacío. Si THRE=1, el último caracter tomado desde el CPU ha sido transmitido y puede ser tomado otro caracter desde el CPU. Si DR=1, un caracter ha sido recibido por el UART y está listo

para ser tomado por el CPU. Otros bits en este registro se fijan en 1 si ocurren errores de: sobrecarga (bit 1), de paridad (bit 2), o por pérdida de secuencia de caracters (bit 3). El bit 4 indica que ha ocurrido una interrupción de ruptura. El bit 6 indica cuando el registro de transmisión del UART está vacío. Si este bit es 1, implica que 2 caracteres pueden enviarse, uno que inmediatamente va a este registro para ser enviado por la línea SOUT, y el otro que espera su turno en el registro de transmisión del UART.

La parte principal de un programa de comunicación es su conjunto de rutinas de servicio de interrupción ISR (Interrupt Service Rutine) que procesan interrupciones del UART. El número de rutinas de servicio de interrupción a diseñarse depende sobre todo de la naturaleza de los servicios de comunicación necesarios. Por ejemplo, una simple aplicación, con un emulador de terminal deberá tener por lo menos un ISR para atender las interrupciones generadas cuando un caracter es recibido por el UART. La rutina ISR podrá leer el caracter esde el UART y almacenarlo en cierto espacio de memoria y el programa principal podrá tomar los caracteres directamente desde este espacio de memoria en lugar de hacerlo desde el UART sin necesidad de que el programa de control esté chequeando en todo momento el estado del UART. Puesto que la rutina de servicio de interrupción es la única responsable para la lectura y almacenamiento de los caracteres de entrada, a menos que la velocidad de transferencia sea extremadamente alta, la probabilidad de que

esta pueda perder un caracter es minima, más aún si el espacio para almacenamiento de los caracteres tomados desde el UART es grande.

Si se requiere el envio de datos a alta velocidad, una aplicación puede crear su propia rutina de interrupción que controle el envio de sus datos y que atienda aquella interrupción generada por el UART cuando su registro de transmisión esté vacio.

La salida de caracteres estaria dirigida hacia una porción de memoria y entonces el ISR se encargaría de enviar los caracteres hacia el UART un byte a la vez. Si es necesario, ISR's adicionales pueden también controlar errores reportados por el UART.

Un programa de comunicación bien diseñado, ejecutándose bajo DOS, puede alcanzar velocidades de transferencia de datos superior a los 9600 bps aún sobre un PC estándar que opere a 4.77 MHZ, demostrado por las diversas aplicaciones comerciales que han sido diseñadas para la transferencia de información entre computadores.

Para accesar y programar un registro interno del UART, primero se debe obtener la dirección base correcta desde la tabla almacenada en el área de datos reservada por el BIOS. Para escribir a los registros internos listados anteriormente, es necesario agregar el correspondiente valor relativo a la dirección base. El rango de los servicios que suministra cada registro del UART es detallado en la tabla

GENERACION E IDENTIFICACION DE LAS INTERRUPCIONES DEL UART

a) Registro de Habilitación de Interrupciones.

BIT

0 = Dato listo para ser tomado desde el registro de recepción.

1 = Registro de transmisión vacio.

2 = Error de ruptura.

3 = Cambio en un pin de entrada RS232

4,5,6 y 7 = Siempre 0

b) Registro de Identificación de Interrupciones.

Los bits 1,2 y 3 identifican la interrupción más reciente. Se puede generar una interrupción cuando:

Bit 321

000 Ocurra un cambio en un pin de entrada RS232.

001 El Registro de Transmisión esté vacio.

100 Exista un caracter en el registro de Recepción.

110 Exista un error de ruptura.

c) Registro de Control de Línea.

BIT

7 = DLAB
6 = Control de Ruptura

0	Activado
1	Desactivado

5,4,3 = Paridad

000	No Paridad
001	Impar
011	Par
101	Mark
111	Space

BIT

2 = Bits de parada

0: 1 Bit
1: 2 Bits

1,0 = Bits de datos

00	5 Bits
01	6 "
10	7 "
11	8 "

d) Registro de Control de Modem.

BIT

7-6-5 = Siempre 0
4 = Reservado
3 = GP02
2 = GP03
1 = RTS
0 = DTR

e) Registro de Estado de la Línea.

BIT

7 = Siempre 0
6 = Registro de TX/RX vacío
5 = Registro de transmisión vacio
4 = Ruptura Detectada
3 = Error de composición del caracter
2 = Error de paridad
1 = Error de sobrecarga
0 = Dato listo en el registro de recepción.

f) Registro de Estado del Modem.

BIT		BIT	
7 =	DCD activado	3 =	Delta DCD
6 =	RI "	2 =	" RI
5 =	DSR "	1 =	" RSR
4 =	CTS "	0 =	" CTS

Tabla 4.2: Rango total de los servicios para la programación directa del UART, necesario para el manejo de interrupciones en la comunicación serial.

Hemos visto como un UART convierte en un computador datos paralelos al formato serial; sin embargo, las salidas TTL de un UART tal como el 8250 no pueden transmitir datos libres de errores a cualquier distancia. La salida serial del 8250 no tiene un nivel de voltaje adecuado, y en cualquier evento, los niveles de voltaje TTL, no son suficientemente inmunes al

ruido para operar adecuadamente a distancias más allá de unos cuantos metros. Para enviar información a cierta distancia considerable, las señales TTL deben ser convertidas a otro formato. Las dos formas más comunes son: la RS-232C, la misma que usa una convención de cierto nivel de voltaje, y la de Lazo de Corriente, la cual se utiliza desde hace algunos años en telegrafía. A velocidades de transmisión relativamente bajas, las señales de lazo de corriente pueden ser transmitidas a grandes distancias (de un pais a otro), es por tal motivo que el lazo de corriente se utiliza en telegrafía. Además la señal de lazo de corriente puede implementarse fácilmente usando optoaisladores, lo cual aisla al equipo transmisor de cualquier daño posterior. La convención RS-232 puede usarse a distancias de hasta 50 pies, pero en la práctica puede ir por lo menos hasta los 100 pies a 9600 bauds. La convención RS-232C a menudo se usa para connección de terminal a modem o terminal a computador y en general es más utilizada que la convención de lazo de corriente. Daremos una explicación de la forma como funciona la convención de lazo de corriente.

La figura 4.1 muestra un circuito que conecta la línea de salida serial (SOUT) de un UART a la línea de entrada serial (SIN) de otro UART a cierta distancia. Ambos UART's se encuentran optoaislados por la conección de lazo de corriente con su propia fuente de corriente. La fuente de voltaje y el resistor han sido selecionados para mantener cerca de 20 mA circulando en el lazo cuando el transmisor (TXD en la figura)

esté en nivel alto. Un nivel bajo es representado por la ausencia de corriente. Por lo tanto, un 1 es representado por un periodo de un flujo de corriente de 20 mA y un 0 por un periodo de ausencia de corriente. Para una operación full dúplex, se necesitan dos lazos de corriente.

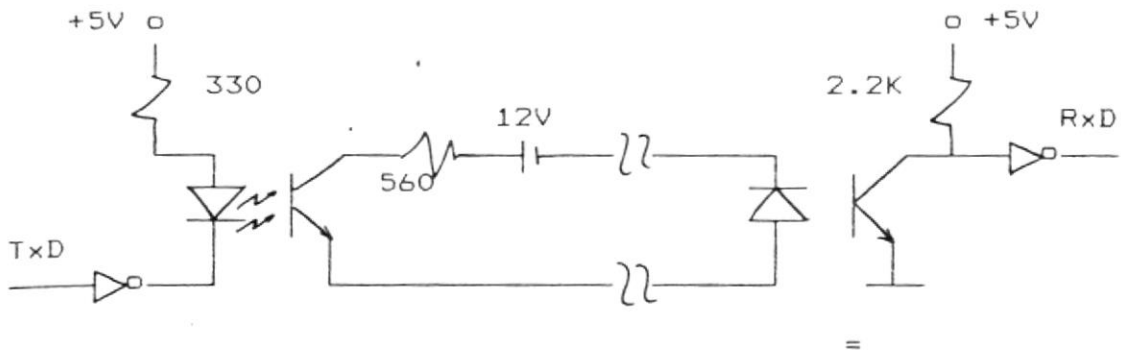


Fig. 4.1: Transmisión Serial usando el método de Lazo de Corriente.

El método de comunicación serial más común es una convención de niveles de voltaje llamado RS-232, el cual es un estándar fijado por la Asociación de Industrias Electrónicas y representa los 1's por -3 a -20 voltios y los 0's por +3 a +20 voltios. El estándar RS-232 también define un conector estándar para comunicación serial, denominado conector DB25. El conector estándar soporta las líneas de control DTR, DSR, RTS y CTS. las mismas que fueron explicadas en el capítulo 1.

Normalmente se utilizan dos circuitos integrados para realizar la conversión de los niveles TTL a señales de nivel RS232-C o viceversa. El integrado MC1488 que internamente contiene 3 compuertas NAND y un inversor, el cual permite convertir señales TTL a señales RS232-C; mientras que por otro lado tenemos el integrado MC1489 que posee 4 inversores

y se utiliza para transformar señales RS232-C a señales TTL.

La figura 4.2 muestra una parte de una interfase RS232-C que soporta las señales de envío y recepción de información (TxD y RxD), y las señales de control (DTR, DSR, RTS y CTS). Note que los integrados MC1488 y MC1489 invierten sus señales de entrada, por lo que todas las señales vistas desde el cableado se encuentran invertidas con respecto a las señales del UART. Nótese también que esta interfase funciona con o sin protocolo de transmisión, puesto que las lineas de control de entrada están preactivadas. Muchas interfases operan de esta forma, por lo que solo requieren la conección de los pines 2, 3 y 7.

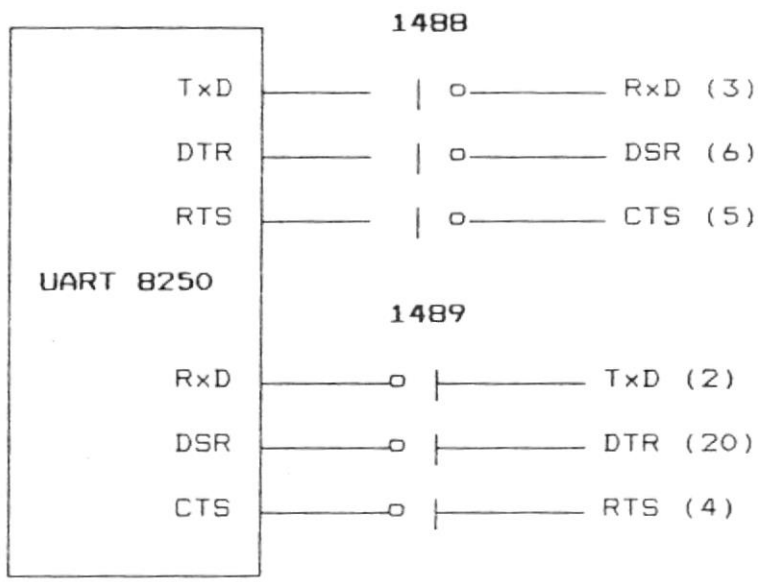


Fig. 4.2: Interfase RS232-C soportando las lineas de control DTR, DSR, RTS, y CTS.

4.3 LA INTERFASE DE TELEX

Hemos descrito el funcionamiento y características del sistema y la de los principales componentes utilizados.

Procederemos ahora a describir el diseño de la tarjeta de control que permite intercomunicar las línea de télex con la interfase serial del computador.

La figura 4.3 muestra la circuitería con las correspondientes señales de envío, recepción y control de transmisión. La señal ENVIO, que procede desde el pin #2 y la señal RECEPCION que se conecta al pin #3 del puerto serial del computador, como su nombre lo indica, sirven para el envío y recepción de información desde y hacia el puerto serial del computador. En condiciones normales, ambas señales se encuentran a un nivel 0.

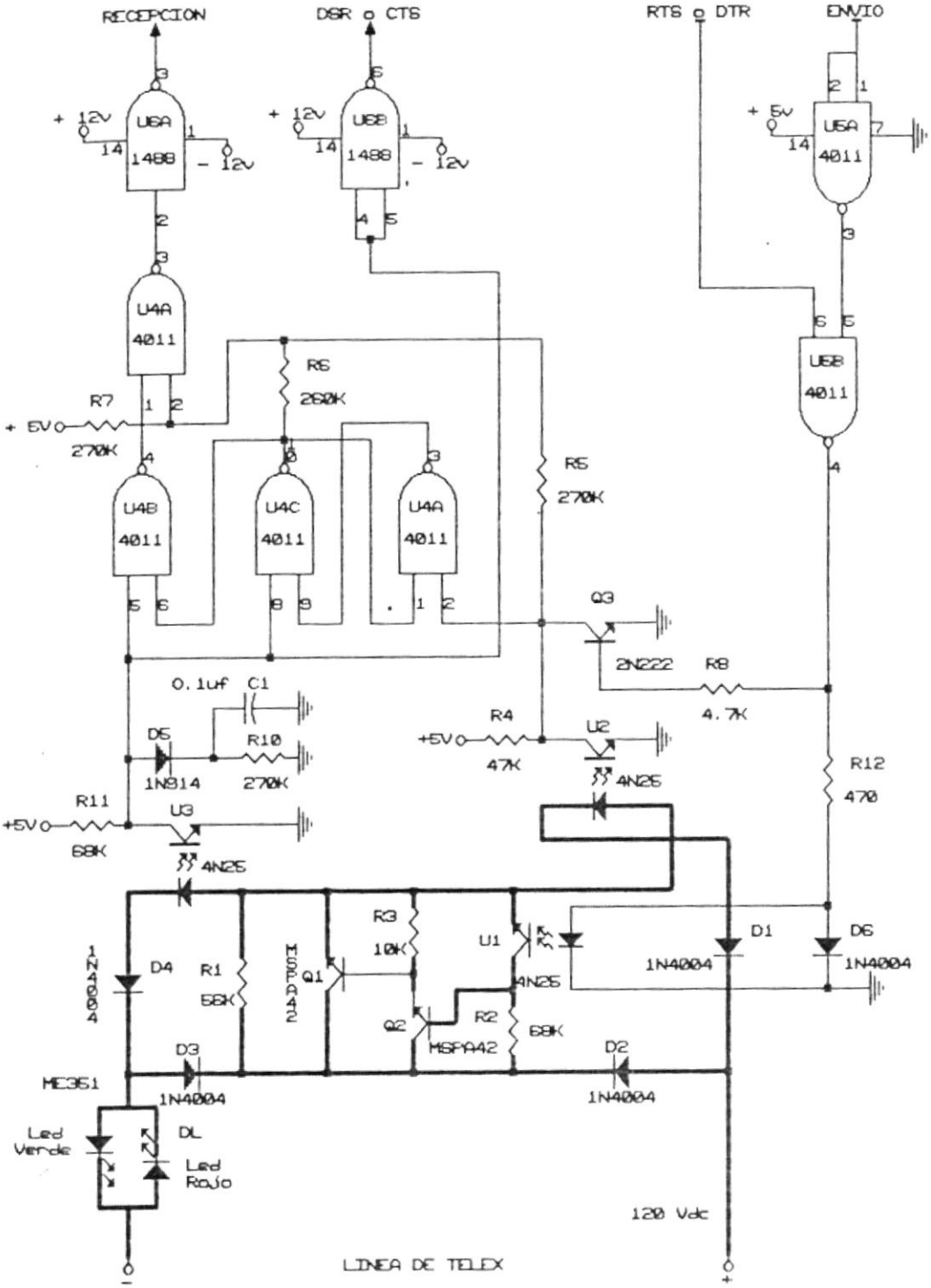
Por otro lado, tenemos las señales RTS y DSR que sirven para el control de envío de información. Estas señales se conectan a los pines 4 y 5 del puerto serial respectivamente, y en condiciones normales, la señal RTS, vista desde el pin #4 del puerto serial tiene un nivel 0, mientras que, debido a la circuitería, la señal DSR se encuentra a un nivel 1.

En la parte izquierda de la circuitería podemos observar la conección de la línea de télex a través de una red de diodos, resistencias y transistores. Esta red se encuentra separada del circuito de control por intermedio de tres integrados de optoaislamiento (U1, U2 y U3 en la figura). U1, como veremos posteriormente se utiliza para invertir la línea de télex; U2 para la recepción de mensajes y U3 para chequear cuando la línea ha invertido su voltaje.

Ahora se describirán tres estados a los que puede operar esta

INTERFASE PARA LA TRANSMISION Y RECEPCION DE MENSAJES DE TELEX

120V/40mA



interfase: Estado de reposo (cuando no existe ni transmisión ni recepción), estado en el que la línea de télex invierte el sentido de la corriente, y el estado en el que se realiza el envío y recepción de informción.

Como se describió en el capítulo 1, las características de funcionamiento de una línea de télex a corriente simplex son:

- Velocidad de transmisión 50, 75, 100 baudios.
- Voltaje telegráfico 120 VDC +/- 10 por ciento
- Corriente de línea en estado de reposo 0 a 5 mA
- Corriente de línea en estado de trabajo 40mA, ajustable.

Pues bien, en estado de reposo ambas señales: ENVIO y RTS se encuentran a un nivel 0 lógico, presentando un nivel 1 a la salida del pin #4, de la compuerta U5B del MC4011, compuesto de cuatro compuertas NAND. Este nivel alto permite la conducción del transistor Q3, presentando un nivel bajo en su colector, lo cual hace que la señal de salida en el pin #3 del integrado MC1488 muestre un nivel bajo, y que corresponde a la señal RECEPCION.

Esta misma señal presente en el pin #4 de la compuerta U5B y que de ahora en adelante la denominaremos SEND, polariza de manera directa el diodo del optoaislador U1 y de acuerdo al sentido de la corriente de la línea de télex, como se muestra en la figura el transistor de U1 conduce. Esto hace que el voltaje presente en la base-emisor del transistor Q2 sea muy pequeño, llevándolo a corte y de manera inmediata el transistor Q1 caerá tambien en la zona de corte debido a que

su unión base-emisor no presenta una diferencia de voltaje por la ausencia de corriente a través de la resistencia R3.

Esto hace que la corriente de línea circule a través del diodo D2, la resistencia R1, el diodo del optoaislador U3 (polarizando su transistor), el diodo D4 y el diodo emisor de luz DL. En estas condiciones, la corriente de línea en estado de reposo esta limitada principalmente por la resistencia R1, que por diferencia de voltaje se puede calcular como:

$$Inormal = \frac{Vdc - Vdiodos}{R1} = \frac{120 - 4 \times 0.7}{56\ K} = 2\ mA$$

Puesto que el transistor del optoaislador U3 está en conducción, se presenta un nivel bajo a la entrada de la compuerta U6B del MC1488, con lo que su salida invertida, perteneciente a la señal DSR presenta un nivel alto. Entonces, el programa de control debe chequear este nivel para determinar el estado de la línea através de esta señal DSR.

Cuando se requiere realizar el envío de transmisión es necesario lograr que la línea de télex invierta su voltaje. Para lograr esto, el programa de control debe fijar un nivel alto en la señal RTS. Esto hace que la compuerta NAND U5B tenga sus dos entradas en nivel alto, con lo que presentará en su salida un nivel bajo, lo que hará que el transistor Q3 deje de conducir, a igual que el diodo D6 y el diodo del optoaislador U1. En estas circunstancias la unión base-emisor del transistor Q2 presenta una diferencia de voltaje

permitiendo su conducción, y llevando a saturación el transistor Q1. Ahora la corriente de línea circulará a través del diodo D2, de R2, R3, Q1, Q2, el diodo de U3, D4 y DL, siendo esta una corriente elevada puesto que R1 prácticamente queda cortocircuitada por Q1. De acuerdo a los valores de las resistencias seleccionadas, esta corriente alcanza los 42 mA, por lo que la central de télex invierte el voltaje de línea creando una nueva ruta de retorno de la corriente.

Si el voltaje de la línea se invierte, la corriente circulará por DL, D3, Q1, Q2, R2, R3 (D2 queda inversamente polarizado, lo mismo que D4 y el dido de U3) y el diodo del optoaislador U2 conduce, activando su transistor.

Ahora el programa de control debe chequear si la línea fue invertida. Para esto, se detecta el nivel de la señal DSR que debe cambiar de nivel alto a nivel bajo. Esto se debe a que si la línea invirtió su voltaje, U3 debe dejar de conducir, con lo cual aparecerá un nivel alto en el colector del transistor y debido al inversor U6B del 1488 la señal DSR quedará a nivel bajo. Con esto, el programa de control podrá estar seguro que se realizó la inversión de la corriente de línea para poder realizar el correspondiente envío de información, pero únicamente después de que haya recibido desde la central de télex la fecha y hora local.

Inmediátamente después que la central de télex invierte el voltaje de línea, ésta comienza a enviar la fecha y hora local a la velocidad de transmisión de 50 bauds y con el

código Baudot. Para representar los 1's y los 0's, la central genera una secuencia de pulsos permitiendo la presencia o ausencia de corriente. Esta secuencia de pulsos se refleja en el transistor de U2 y, dado que Q3 está en corte, la información pasa directamente a la señal RECEPCION y por consiguente el programa de control puede recibirla.

Al final de la transmisión de la fecha y hora local, la central envia la secuencia GA (Go Ahead) para indicar al abonado que prosiga adelante. A partir de este momento el programa de control podrá enviar información a través del pin #2 del puerto serial del computador. Esto creará una secuencia de pulsos a la salida de la compuerta NAND U5B, secuencia que hará que el optoaislador U1 oscile llevando a corte y saturación al transistor Q1, haciendo que la corriente que circule por la línea de télex varie de 2 mA a 42 mA.

Existen dos formas por las que puede ocurrir un final de comunicación: O porque la central de télex invirtió la línea (a su estado normal) por orden del otro abonado, o porque la invierte por orden local. Para esto, el programa de control debe estar siempre chequeando el estado de la señal DSR, porque si esta señal cambia a nivel alto, quiere decir que la central invirtió el voltaje a su estado normal y por lo tanto el programa debe fijar a nivel bajo la señal RTS, quedando la linea en reposo. Para cortar la comunicación desde el programa de control solo basta con enviar un nivel bajo en la señal RTS para que la línea vuelva a su posición inicial.

Otra forma de cortar la comunicación es enviando por la línea cuatro veces la letra m o una secuencia de puntos (....) y en este caso se produce el mismo primer efecto. Siempre que se desee cortar la comunicación desde el programa, éste debe chequear la señal DSR para estar seguro del final del enlace.

Si en condiciones normales, la línea se invierte por la llegada de un mensaje, el programa puede detectar esto al recibir el cambio de nivel alto a nivel bajo de la señal DSR. Si esto sucede, el programa debe fijar en nivel alto la señal RTS para permitir el ingreso de información. Esto es importante porque si RTS no se fija en nivel alto, la salida de la compuerta U5B estará en nivel alto facilitando la conducción del transistor Q3, el cual no permitirá el ingreso de información.

Además, puede suceder que por algunos minutos la línea permanezca invertida sin que se reciba ni envie información. Para esto se debe crear un lazo de hasta 3 minutos, tiempo máximo durante el cual la línea puede estar sin uso; caso contrario debe cortarse el enlace automáticamente. Si bien es cierto el abonado local sabe que siempre debe finalizar su comunicación, no podemos asegurar que un abonado que llame deje la línea abierta por demasiado tiempo. Es por esto el fin de comunicación automático.

Establecidas las funciones del sistema, así como sus entradas y salidas, el siguiente capítulo describe en detalle el programa necesario para el control de la interfase.

CAPITULO 5

ELABORACION DEL PROGRAMA PARA EL ENVIO, RECEPCION E IMPRESION DE MENSAJES DE TELEX UTILIZANDO EL LENGUAJE ENSAMBLADOR DEL MICROPROCESADOR 8088

5.1 INTRODUCCION

Los programas que permanecen residentes en memoria, conocidos como TSR (Terminate and Stay Resident), representan la vía más segura para realizar una tarea detrás de lo que significa un Sistema Operativo diseñado para procesar un trabajo a la vez. Son programas que se ejecutan una sola vez desde la línea de comandos del DOS, reservan cierta porción de memoria para almacenamiento de su propio código y devuelven en control al DOS, luego de lo cual residen pacientemente en memoria esperando por cierta secuencia de teclas antes de poder activarse y ejecutar ciertas acciones, suspendiendo cualquier operación que algún programa no residente haya estado realizando.

TELEX, es un ejemplo clásico de un programa que se une al sistema operativo a través de la función 31h de la interrupción 21h del DOS y permite activar una ventana en cualquier momento (bueno, casi en cualquier momento), aún

desde dentro de la ejecución de un programa de aplicación, e inspeccionar a través del contenido de los directorios o subdirectorios de cualquiera de las unidades de disco del PC; así como leer ,redactar, grabar, o imprimir cualquier documento. Mas aún, TELEX permite que un documento pueda ser enviado o recibido por la línea de comunicación del puerto serial del computador hacia una línea de télex conectada al PC a través de una circuitería especial, grabando los mensajes recibidos como archivos hacia cierta unidad de disco especificada en la instalación del programa.

Sin embargo, para realizar todo esto, TELEX debe utilizar los servicios del DOS de una manera muy diferente de la que usan muchos programas residentes. Muchas de las técnicas usadas en programas excelentes tales como SIDEKICK están presentes en TELEX.

TELEX debe ser ejecutado una vez al inicio de una sesión o a través del archivo AUTOEXEC.BAT. Luego, presionando la secuencia ALT-SHIFT IZQ se activará una ventana en la cual se muestran todas las opciones que TELEX ofrece. Además, TELEX puede coexistir tranquilamente con muchas aplicaciones, incluyendo otros programas de su misma especie.

5.2 CARACTERISTICAS DE LOS PROGRAMAS RESIDENTES.

Antes de examinar el programa TELEX en detalles es necesario considerar algunas de las características de los programas que terminan y permanecen residentes en memoria. Básicamente existen dos tipos de programas TSR. La mayoria de ellos no

usa los servicios extendidos de entrada/salida de archivos del DOS; los más complejos si lo hacen.

Los del primer tipo son relativamente sencillos; típicamente son programas que toman el control de la interrupción del teclado y residen en memoria esperando por cierta combinación de teclas para tomar el control del sistema y activarse dejando congelada cualquier tarea que se estaba realizando.

Normalmente estos programa residentes son elaborados de una manera tal que toman el control del sistema casi a ciegas, sin primero asegurarse de que alguna situación crítica del mismo esté presente.

Hasta cierto punto esto está bien. Por lo general, un programador asume que el usuario no tratará de activar su aplicación residente en medio de, digamos, un proceso de formateo o copia de un disco, o simplemente cuando se esté ejecutando el mandato DIR desde la línea de comandos del DOS. Sin embargo, puede suceder tal situación y es precisamente aquí cuando comienzan a surgir las dificultades con los programas residentes cuyo resultado final es casi siempre en la paralización de todo el sistema.

Por eso, estos programas únicamente realizan una pocas tareas, como por ejemplo enviar unos cuantos códigos de control a la impresora , ejecutar cálculos matemáticos o sinó graban el contenido de la memoria de video, la manipulan y luego la recuperan, sin usar en ningún momento

rutinas de acceso al disco para finalmente devolver el control al programa que interrumpieron.

Existen otros programas tales como SIDEKICK, que hacen cada cosa una parte residente, modificando todo un bloque de interrupciones para acceso al disco.Desde el punto de vista de un usuario, lo que diferencia SIDEKICK de otros programas residentes es que éste lee y escribe al disco tal como si se tratase de una aplicación no residente. Desde una perspectiva de programación, esto significa que SIDEKICK hace un uso restringido de los servicios de la interrupción 21h del DOS.

A menos de que se tomen algunas precauciones, los servicios del DOS suministrados a través las interrupciones 21h, 25h y 26h no pueden ser usados desde un TSR. Al usar uno de ellos se encontrarán todas las probabilidades de destruir un sistema.

Existe un par de preguntas por contestar antes de entrar a una explicación de cómo pueden ser usadas las rutinas del DOS desde un programa residente. La primera es: ¿ Porqué los servicios de las interrupciones del DOS pueden ser libremente ejecutados desde una aplicación normal, y porqué no desde dentro de un programa residente ?. La razón es que los servicios del DOS son **NO-REENTRANTES**. ¿ Qué significa esto?. Esto significa que una vez que una función comienza, esta debe continuar hasta finalizar su proceso antes de que otra función pueda comenzar. Por lo

tanto, si un servicio del DOS se está ejecutando, éste no puede ser interrumpido para dar paso a la ejecución de otro.

Por supuesto, un programa normal nunca intenta llamar a algún servicio del DOS desde dentro de otro, porque una vez que se llama a una interrupción 21h, el DOS toma el control del sistema, y ningún otro llamado puede hacerse hasta que el DOS termine su tarea. Pero, si un programa residente interrumpe un servicio del DOS, (normalmente un TSR puede ser activado en cualquier momento) y entonces él mismo usa una interrupción 21h, se estaría aplicando el problema de reentrada.

Usualmente, el problema no se observa sinó hasta que la segunda llamada finalice su proceso y el programa residente trate de devolver el control al servicio del DOS previamente interrumpido. Por consiguiente, cualquier programa residente que desee realizar llamadas a los servicios de la interrupción 21h del DOS primero debe asegurarse de no interferir con la ejecución de otro.

La segunda pregunta por contestar es justamente, ¿ porque razón las rutinas del DOS son no-reentrantes ?. Lo que sucede es que cuando se realiza un pedido de algún servicio de la interrupción 21h, uno de los primeros pasos que el DOS realiza es conmutar a una de 3 pilas internas que éste contiene. Lo mismo realiza cuando se ejecuta un llamado a las interrupciones 25h y 26h.

El cambio de pila se realiza grabando el contenido de los

registros SS y SP para recuperarlos a la salida y luego cambiándolos para que apunten a un área de memoria reservada dentro de un espacio de trabajo del sistema operativo. Si un servicio del DOS es interrumpido por otro, todo lo que éste tenga almacenado en la pila será borrado por el otro.Esto es, el contenido de los registros SS y SP del primer servicio será reemplazado por el contenido de los registros del otro servicio. Otra de las cosas igualmente a destruirse es la dirección de retorno de la rutina del servicio interrumpido, la cual fue almacenada en la pila por el microprocesador. De tal manera que cuando esta rutina ejecute un IRET, el control será direccionado a algún lugar aleatorio de la memoria por lo que el comportamiento del sistema será totalmente impredecible.

La pila interna que use el DOS depende del número de interrupción, y, para la interrupción 21h, del número de la función. Las interrupciones 25h y 26h siempre usan la misma pila. Las funciones 1 hasta la 0Ch de la interrupción 21h (incluyendo unas cuantas funciones de orden superior) usan otra pila; mientras que, las funciones 0, 0Dh y superiores a esta (con algunas cuantas excepciones), usan la misma pila empleada por las interrupciones 25h y 26h.

Por el contrario, las rutinas del BIOS son REENTRANTES porque ellas no emplean una pila especial. En realidad, ellas asumen que el programa que las invocó tiene un espacio de pila suficiente como para mantener el guardado y vaciado de unas cuantas palabras. En casi todos los casos, esta es una

suposición segura. Sin embargo, muchos programas residentes usan su propia pila para estar absolutamente seguros que sus requerimientos de pila serán suficientes.

5.3 SERVICIOS DE ACCESO AL DOS DESDE UN PROGRAMA RESIDENTE

Como hemos visto, para que un programa residente pueda usar los servicios del DOS, es necesario que la rutina del programa residente no tome el control cuando alguna función de la interrupción 21h, 25h o 26h está siendo procesada. Una de las maneras más sencillas de evitar tales circunstancias es interceptar los llamados a esas interrupciones y fijar una bandera para indicar que un servicio del DOS se está ejecutando y borrar esa bandera a su salida. De modo que cada vez que el programa residente intente activarse, éste pueda chequear esa bandera y desistir de sus propositos si encuentra que ésta tiene un valor igual a 1. Esta es una manera fácil de evitar las condiciones de reentrada.

Sin embargo, el DOS contiene internamente una bandera, denominada INDOS que cualquier programa puede accesar para ver si alguna rutina del DOS está procesandose.

La dirección de esa bandera se la puede obtener llamando a la función 34h de la interrupción 21h. Esta función retorna la dirección del segmento en el registro ES y su desplazamiento en el registro BX. Chequeando el byte de esta bandera dentro del área de trabajo del sistema operativo y antes de activarse, un programa residente puede asegurarse

de que un servicio del DOS no afectará a otro.

Si buscamos alguna documentación sobre la función 34h del DOS en el manual de referencias técnicas (o en casi cualquier manual de programación), veremos que la única información proporcionada es : " Usada internamente por el DOS". Esta es una de las causas del poco conocimiento de como usar los servicios del DOS desde un TSR.

Existe un problema ante todo lo especificado. Si hemos dicho que un programa residente no debe activarse cuando una rutina de interrupción del DOS se está ejecutando, ¿ que pasaría si un programa de este tipo requiere acceso durante la ejecución de una función prolongada del DOS, tal como la función 0Ah, la misma que lee una linea de texto desde el teclado ?. Realmente este es un problema, porque la función 0Ah está corriendo cuando el DOS está en espera de la entrada de una linea de comando en el prompt A>, en cuyo caso un programa residente jamás podrá activarse mientras el DOS tenga el control del sistema

Aparentemente las cosas se ponen difíciles, pero, al analizar la rutina de la función 0Ah, vemos que mientras esta función espera por el ingreso desde el teclado, esta misma función está llamando continuamente a la interrupción 28h (otra interrupción del DOS no documentada, vital para la programación de programas residentes avanzados), la misma que puede ser interceptada en cualquier momento.

Cuando se realiza una llamada a la interrupción

28h, únicamente las funciones del DOS de orden mayor que la 0Ch pueden ser libremente invocadas sin causar caida alguna del sistema aún cuando la bandera del DOS esté puesta a 1. Así, un programa TSR puede preparar su propia rutina manipuladora de la interrupción 28h y llamar a alguna función de orden superior a la 0Ch desde dentro de ésta. El porque nosotros podemos usar solo servicios de las funciones mayores que la 0Ch es contestado recordando como está estructurado el DOS. De las 3 pilas internas que el DOS mantiene, una es usada casi solamente para los llamados a los servicios 1-0Ch. La interrupción 28h es ejecutada solo por las rutinas del DOS que están en ese rango. Entonces, si un programa residente es activado por intermedio de esta interrupción, debe evitar llamar a cualquiera de esas funciones.

Los programas residentes podrán usar los servicios de disco y otras funciones si se toman las precauciones anotadas. Pero, existen ocasiones en las que tales programas no pueden activarse. Una de estas es por ejemplo cuando usamos el comando TYPE del DOS para listar un archivo grande, y luego tratamos de activar un programa residente tal como SIDEKICK en la mitad del listado. Podemos observar que SIDEKICK no se activa. El indicador INDOS del DOS está fijado 1 y la interrupción 28h no es ejecutada, asi que una rutina residente tal como SIDEKICK que realiza operaciones de entrada y salida de archivos no podrá interrumpir al comando TYPE. Como vemos, el sistema no es perfecto, pero al menos es

la mejor solución que puede encontrarse hasta que se desarrollen versiones del DOS que permitan operaciones de multitareas.

5.4 CONSIDERACIONES DE INTERRUPCION

Muchos programas residentes necesitan redireccionar ciertos vectores de interrupción para poder activarse a través de estos. La tabla de los vectores de interrupción está localizada en los primeros 1024 bytes de la memoria del sistema, y contiene los punteros de todos los vectores de interrupción usados por el procesador. Se necesitan 4 bytes para cada puntero: 2 para la dirección de segmento y 2 para su desplazamiento. Redireccionando los vectores seleccionados es lo que da a los programas residentes la habilidad de monitorear el teclado o cualquier otra operación interna. Normalmente, cuando un vector de interrupción es interceptado, el vector original es almacenado de tal manera que el programa residente pueda posteriormente saltar a la dirección contenida en el vector antiguo. Esta técnica permite a los programas residentes ser transparentes para el resto del sistema.

Además de usar las características de no-reentrada del DOS, un programa residente que accese al disco, deberá también asegurarse de no interferir con el programa de aplicación que esté interrumpiendo. Toda aplicación, residente o no que lea información del disco, debe reservar una porción de memoria (o usar el área que por defecto

suministra el DOS) para usarla como Area de Transferencia del Disco o DTA (Disk Transfer Area). Esta porción de memoria es utilizada por el DOS para almacenar los datos correspondientes a todas las lecturas y escrituras de archivos realizadas con el conjunto de llamadas tradicionales

Esta área puede estar en cualquier porción de la memoria y puede ser establecida por los porgramas llamando a la función 1Ah de la interrupción 21h del DOS. Un programa residente debe definir su propio DTA cada vez que se active, guardando previamente la dirección del DTA original y recuperándola a su salida.

Otra consideración importante pero muchas veces pasada por alto es el hecho de qué acción se debe tomar en el caso de que ocurra un Error Crítico. Un error crítico es un tipo de error especial que detecta el DOS en situaciones de emergencia, tales como al tratar de grabar o leer hacia o desde un disco defectuoso o cuando tratamos de imprimir un documento sin tener lista la impresora. En tales situaciones, el DOS automáticamente genera una Interrupción Manipuladora de Errores Críticos. Esta es la interrupción 24h y es la que presenta el famoso mensaje: 'Cancelar, Reintentar o Ignorar'.

Además, la dirección asociada con la interrupción 23h señala a la rutina de manipulación que será llamada siempre que el DOS responda a un acción de ruptura desde el teclado. La clave de ruptura se genera en un teclado PC estándar mediante Ctrl-Break,o en cualquier teclado mediante Ctrl-C.

La respuesta que por defecto suministra el DOS en la interrupción de ruptura es finalizar el programa o el fichero de proceso por lotes que está siendo ejecutado.Si nuestros programas utilizan su propio manipulador de interrupción de ruptura, pueden hacer que el DOS tome cualquier acción que ellos deseen sin importar lo extensa o compleja que sea.

Es muy importante que los programas residentes empleen un manipulador de errores críticos propio para prevenir desde éste cualquier condición de error posterior, porque la falla de una rutina residente seguramente bloqueará el programa suspendido.

Un programa residente completo que realize operaciones de entrada/salida de archivos, debe por consiguiente a su ingreso cambiar los vectores de interrupción 23h y 24h a rutinas propias y asegurarse de restablecer estos vectores a sus valores iniciales a su salida.

Por otro lado, la interrupción 13h del BIOS, suministra todos los servicios de disco de bajo nivel; para lectura, escritura y formateo individual de sectores y cilindros. En un sentido común, esos servicios son NO-REENTRANTES, tal como los servicios de la interrupción 21h del DOS, pero por una razón diferente.

No existen problemas con localizaciones de pila, de hecho, como habíamos dicho, las rutinas del BIOS no usan una pila interna. Sin embargo, existe el peligro de interrumpir el

movimiento de la cabeza lectora de la unidad de disco desde un cilindro a otro. Si la rutina residente realiza cualquier operación de entrada/salida de archivos, la cabeza lectora será movida de su posición en la que se encontraba cuando fue interrumpida y no será retornada a su posición inicial cuando el programa residente devuelva el control a la aplicación que se estaba ejecutando. Entonces, es necesario que un programa residente de las características anotadas no se active cuando se están realizando acceso al disco y debe ser prevenido mediante un indicador que refleje la ejecución de uno de los servicios de la interrupción 13h.

5.5 CONSIDERACIONES DE VIDEO

Todo programa residente o no que realice escrituras directas de bloques de caracteres en la memoria de video debe tomar ciertas precauciones especiales cuando el sistema esté usando un adaptador de color/gráficos estándar.

Existen 6 tipos clásicos de adaptadores de video de uso común: el adaptador de video monocromático o MDA, el adaptador de color/gráficos o CGA, el adaptador de gráficos ampliado EGA para monitor monocromático, el adaptador de gráficos ampliado EGA para color, el MCGA incluido en el nuevo modelo de computadores PS modelo 30 y el nuevo adaptador de video gráfico VGA. Cada uno de esos adaptadores puede soportar ciertos modos de video. Los textos sobre un adaptador monocromático siempre usan el segmento B000h y los textos en uno de color usan el segmento B800h. Cada caracter

en la memoria de video usa dos bytes: el primer byte almacena el valor ASCII del caracter mostrado y el segundo byte contiene el atributo de color, intensidad, subrayado o parpadeo del caracter.

Como hemos visto, el segmento de video que use el computador dependará del tipo de adaptador de video que esté usando (sea B000h para monocromático o B800h para color). En todo caso, esta memoria de video siempre está localizada en el adaptador de video, no en la memoria instalada en la placa principal del sistema. La circuitería de video lee automáticamente sesenta veces por segundo esa memoria y envía su contenido hacia el barrido del rayo dentro del monitor. Cuando decimos que el 8088 escribe o lee diréctamente hacia o desde la memoria de video, puede ocurrir un conflicto si el 8088 y la circuitería de video tratan de accesar a la misma localización de memoria al mismo tiempo, cuyo resultado es una desagradable interferencia sobre la pantalla. Este inconveniente se observa únicamente en los adaptadores de color CGA. El adaptador EGA usa un diseño especial, tal que la sincronización para los requerimientos de lecturas y escrituras simultáneas son gobernadas automáticamente por la circuitería del adaptador.

Cuando se escriba o lea en la memoria de un adaptador de color estándar CGA, es necesario que los programas accesen a esta memoria solo durante los períodos de tiempo en los cuales la circuitería no lo está haciendo. La forma de realizar esto es sensar el bit de retrazo horizontal en el

puerto de estado de video. Un monitor de video crea las imágenes moviendo repetidamente un rayo de electrones desde la izquierda hacia la derecha a través de la pantalla bajando una línea en cada barrido secuencial. Al final de cada línea, el rayo es apagado por un corto período de tiempo durante el cual éste regresa al lado izquierdo para enviar la siguiente línea (podemos ver estos retrazos si aumentamos el brillo en el monitor). Este punto es importante, puesto que cuando el adaptador de video está realizando un retrazo horizontal, la señal de salida de video es desactivada, dejando tiempo libre para realizar escrituras directas a su memoria. El inconveniente es que este tiempo de retrazo es extremadamente pequeño (unos pocos microsegundos), de tal forma que solo existe tiempo para escribir uno o dos bytes. Sin embargo, este retrazo es tan frecuente que podemos esperar por el siguiente retrazo para escribir cada caracter diréctamente a la memoria de video.

Este método es mucho más rápido que usar las rutinas del BIOS.

El BIOS usa un método similar y por supuesto también espera por un retrazo horizontal, pero es lento por varios aspectos: Cada vez que se realiza un llamado a una rutina del BIOS, éste primero debe chequear qué función del BIOS se está llamando a ejecutar. Si la función del BIOS es mostrar un caracter en la pantalla, éste tiene que determinar qué tipo de adaptador de video está presente en el sistema. Si el adaptador es de color CGA, el BIOS debe chequear en qué modo

de video está operando (80 o 40 columnas). Luego, tiene que encontrar dónde está localizado el cursor y calcular su posición para escribir el caracter. Solamente en este momento podrá esperar por el retrazo horizontal para finalmente mostrar el caracter en la pantalla. Todo esto se realiza cada vez que el BIOS muestra un caracter. ¿No es demasiado lento?.

Entonces, todo el trabajo previo se lo puede realizar una sola vez en el programa. El tipo de adaptador, y la dirección del segmento de video pueden determinarse al inicio del programa y entonces un simple lazo puede realizar el chequeo del retrazo horizontal y luego ejecutar las operaciones de escritura.

Existe también un período de retrazo vertical y este es mucho más largo que el horizontal (cerca de 17 milisegundos), suficiente como para almacenar una línea entera de caracteres o más durante el retrazo vertical aunque el número exacto varía con la velocidad del PC. Es muy poco utilizado.

5.6 DENTRO DEL PROGRAMA TELEX

Conociendo la teoría anterior, el método es una condición necesaria pero no suficiente para implementarlo. Aún existen muchos detalles que deben ser tomados en cuenta; desde cómo coordinar intercepciones de interrupciones simultáneas, hasta como hacer un programa residente compatible con otro. Una forma de entender esto es observando dentro de un programa que considera todos estos detalles, tal como TELEX.

TELEX es un programa residente y por lo tanto necesita ser cargado permanentemente a memoria antes de que éste pueda ser usado. Una vez que TELEX es instalado, se puede activar el programa usando la secuencia ALT SHIFT IZQ.

Al igual que todas las subroutinas residentes, TELEX comienza ejecutando una sección de código que prepara el escenario para la parte del programa que permanecerá residente después de su instalación. Una vez completada esta tarea, el código de inicialización es eliminado cuando el espacio de memoria que éste ocupa sea devuelto al DOS.

El procedimiento de inicialización es ejecutado cuando el programa es cargado a memoria. Su tarea principal es modificar la tabla de los vectores de interrupción de tal forma que las interrupciones 05h, 08h, 09h, 10h, 13h, 1Ch, y 28h sean redireccionadas a subroutinas propias de TELEX, sin que la función básica que éstas realizan sea afectada.

Otra tarea de la etapa de inicialización es realizar una búsqueda rápida de una copia del mismo programa previamente cargado a memoria. Si tratamos de cargar a mamoria una segunda copia de TELEX, el programa mostrará el mensaje '**El programa ya está residente en memoria ...**' y simplemente sale. Esto evitará tener en memoria múltiples copias del mismo programa residente.

TELEX usa llamados a ciertas funciones o manipuladores de interrupciones del DOS disponibles a partir de las versiones

3.0 (por ejemplo el manipulador de la interrupción 24h), y, usando un llamado a la función 30h del DOS, chequea qué versión del Sistema Operativo está corriendo en el computador. Si está corriendo una versión inferior a la 3.0, TELEX finaliza mostrando el mensaje '**Versión del DOS no soportada**'.

Luego se realiza un llamado a la función 34h del DOS para obtener la dirección de la bandera INDOS que indica cuando el DOS está ocupado ejecutando un proceso. La dirección segmento obtenida en el registro ES es almacenada en SEGMENTO_DOS y la dirección desplazamiento obtenida en BX es almacenada en INDOS. Bueno, ¿ porqué hacer esto en la inicialización y no cuando el programa residente trate de activarse ?. La razón es que si el computador está ejecutando un servicio de la interrupción 21h, un llamado a la función 34h desde el programa residente puede bloquear el sistema. Por lo tanto, la dirección de la bandera del DOS es grabada en una variable del programa para uso posterior durante la fase no residente y esa misma variable es chequeada diréctamente desde la porción residente cuando quiera que TELEX lo requiera. Así mismo se obtiene y almacena en una variable del programa, la dirección del indicador de errores críticos.

TELEX usa la caracteristica de esperar por un retrazo horizontal antes de escribir cualquier caracter en la memoria de video, por lo que necesita saber qué tipo de adaptador de video tiene el sistema. Para esto, TELEX realiza un llamado a la función 12h de la interrupción 10h del BIOS para primero

determinar si existe un adaptador de video tipo EGA, cargando 10h en el registro BL a su entrada. Esta función retornará un 0 un 1 en el mismo registro para indicar que existe un adaptador EGA operando en modo color o en el modo monocromático. Sin embargo, si no existe un adaptador EGA, un llamado a 12h simplemente retorná el registro BL con su mismo valor (10h).

En tal caso, TELEX usa la dirección del controlador de video CRT 40:63h para obtener el tipo de adaptador de video que está instalado, ya sea CGA (color) o MDA (monocromático). Finalmente, TELEX indicará en una variable denominada ADAPTADOR el tipo de adaptador de video instalado, fijando en esa variable un valor 0 si el adaptador es monocromático MDA, 1 si el adaptador es de color estándar CGA o un valor de 2 si el adaptador es EGA. En la variable SGMNTO_DE_VIDEO se almacena uno de los valores, B000h o B800h para indicar el correspondiente segmento de video (monocromático o color) que utilizará TELEX al mostrar sus ventanas. Siempre que TELEX necesite mostrar una secuencia de caracteres en la pantalla chequeará esta variable para ver si el adaptador instalado es de color (ADAPTADOR=1). Si es así, deberá esperar por el retrazo horizontal.

El paso de inicialización final es redireccionar todos los vectores de interrupción que TELEX necesite. El programa intercepta siete interrupciones (sin incluir las interrupciones 1Bh ,23h y 24h, cuyos vectores son manipulados cuando el programa trate de activarse, no durante la

instalación). Cada una de estas interrupciones juega un papel importante en el proceso de activar el programa. Estas son:

- La interrupción de impresión de la pantalla 05h. la misma que actúa cuando son presionadas la teclas SHIFT PRT SCR. Siempre que TELEX esté activado, ignora cualquier intento de impresión de pantalla . La subroutina de control desde el programa se denomina INT05.

- La interrupción del reloj principal del sistema 08h que, bajo circunstancias normales es generada 18 veces por segundo por el integrado de tiempo programble 8253. Interrupción controlada por la subroutina TIMER en el programa de TELEX.

- La interrupción del teclado 09h, (interrupción generada por un microprocesador dedicado dentro del teclado), usada para detectar la secuencia de entrada ALT SHIFT IZQ, que activará el programa. Su subroutina de control se denomina TECLADO.

- La interrupción 28h de retorno de proceso del DOS, la cual bajo condiciones normales permite a los programas residentes pasar por alto el indicador INDOS usado por el DOS cuando está ejecutando un proceso. Controlada por la subroutina BACKPROC (de BACKground PROCess).

- Las interrupciones 10h, y 13h, que proporcionan todos los servicios estándar de video y accesos a disco de la ROM-BIOS respectivamente. Rutinas de control: VIDEO y BIOS_DISCO.

- La interrupción 1Ch generada por la interrupción 8h, y usada por TELEX para mostrar la fecha y hora del sistema,

también para detectar el ingreso de caracteres desde la línea de comunicación, o para grabar los mensajes de télex recibidos. La subroutina de control se denomina INTIC.

Luego de modificar los vectores de interrupción anotados, el programa borra todo el espacio de memoria que permanecerá residente y que utilizará para almacenamiento de textos e inicializa ciertas variables y punteros internos, para finalmente mostrar en la pantalla el siguiente mensaje:

Telex en una PC/XT/AT 1.0

ESPOL-GYE 1990. Por Freddy Pinto Carpio

Secuencia de entrada ALT-SHIFT izq

La siguiente etapa de inicialización del programa es ver si es posible instalar TELEX en uno de los puertos de comunicación serial: COM1 o COM2 para el envío y recepción de mensajes. Las direcciones absolutas para todos los puertos seriales presentes en el sistema están almacenados en el área de datos reservada por el BIOS a partir de la localización 0040:0000. El programa chequea esas localizaciones, y busca por COM1 o COM2. Si no encuentra ninguno de estos puertos se muestra el siguiente mensaje de error: '**Puertos Serie COM1 o COM2 no están presentes en el sistema...**' y el proceso de instalación finaliza, quedando TELEX de todas formas listo para ser activado, pero funcionará en formar local. En el caso de que COM1 o COM2 estuviesen presentes, el primer puerto de comunicación disponible es configurado para que opere a la velocidad de 50 baudios, 5 bits de datos, y 1.5

bits de parada, necesario para la comunicación con la linea de TELEX; después de lo cual el programa muestra el mensaje 'TELEX instalado en COMx' donde x puede ser 1 o 2.

Al final de la instalación, el control es retornado al DOS usando la función 31h de la interrupción 21h, 'Terminar pero permanecer residente'. Para TELEX, el espacio de memoria para mantener residente es todo el segmento del código del programa (a excepción de la porción de inicialización), más la memoria suficiente (unos 25741 bytes) reservada para la manipulación de textos y archivos de disco.

TELEX contiene internamente un byte, denominado INDICADORES en el programa fuente, cuyos bits se usan para los siguientes propósitos:

Bit:

0: Indica que el programa residente de TELEX está activado. Si la interrupción 09h encuentra este bit con un valor 1, ignora el proceso de detectar la secuencia ALT SHIFT IZQ.

1: Indica cuando el BIOS está ejecutando un servicio de video por medio de la interrupción 10h. Este bit es sensado por la interrupciones 8h y 28h.

2: Indica cuando el BIOS está ejecutando un servicio de disco estándar a través de la interrupción 13h. Bit sensado por la interrupciones 8h y 28h.

5: Indica cuando TELEX está ejecutando la subroutina del

DIRECTORIO. Este bit se usa para evitar la salida de reloj a la pantalla mientras la subroutina de directorio de TELEX esté corriendo.

6: Bit fijado por las interrupciones 8h y 28h cuando intentan activar el programa TELEX.

7: Bit fijado por la interrupción de teclado 09h cuando detecte la secuencia ALT SHIFT IZQ, siempre y cuando el bit 0 tenga un valor cero. Si las interrupciones 08h o 28h encuentran este bit con un valor 1, tratan de activar el programa TELEX.

Una vez instalados, muchos programas residentes chequean en todo momento el ingreso de un caracter desde el teclado, detectando la combinación de sus teclas destinadas para indicar cuando deben activarse. En este aspecto, TELEX no es diferente. TELEX intercepta la interrupción 9h que es generada cada vez que una tecla es presionada; pero si TELEX detecta la combinación ALT SHIFT IZQ, la acción que toma es muy diferente a la de muchos programas residentes.

TELEX no puede tomar a ciegas el control del computador y abrir su primera ventana que es la del EDITOR. En lugar de esto, primero debe asegurarse de no estar interrumpiendo ciertas subroutinas de servicio del DOS o algún servicio de disco del BIOS. Si tal situación ocurre, TELEX debe esperar para activarse después. Por lo tanto, la subroutina manipuladora del teclado de TELEX, no abre diréctamente la primera ventana, pero si se encarga de fijar (desde 0 a 1) el

bit 7 de la variable INDICADORES y simplemente sale. Si el programa TELEX está activado, esta subroutina ignora la secuencia ALT-CTRL-DEL.

Las interrupciones 8h y 28h son usadas de una manera coordinada para responder al requerimiento de activar el programa. Su propósito es chequear el bit 7 de INDICADORES (que fue fijado por la subroutina de TECLADO), y si es posible abrir la ventana del EDITOR. La primera subroutina se denomina TIMER, y corresponde a la interrupción 8h. TIMER chequea el bit 7 de INDICADORES y finaliza por medio de un IRET si este bit es igual a cero; pero si el bit está en 1, TIMER chequea los bits indicadores del estado del BIOS, del DOS y el de Error Crítico, los mismos que le indicarán si alguno de sus servicios se está ejecutando. Si estos bits son iguales a cero, TIMER llama al procedimiento PRINCIPAL, desde donde TELEX toma el control del sistema.

La segunda subroutina complementa la tarea de la primera: BACKPROC recibe el control cada vez que una interrupción 28h es generada y de la misma manera que la subroutina TIMER, chequea el bit 7 de INDICADORES. Si este bit es igual a cero, un IRET es ejecutado para finalizar la interrupción.

De esta forma podemos ver como TIMER y BACKPROC coordinan sus responsabilidades chequeando cada una independientemente los bits de INDICADORES, y la primera de ellas que pueda activar el programa abrirá su primera ventana, pero únicamente después de que el bit 7 de INDICADORES sea fijado a cero, de

tal manera de que no se vaya a realizar una doble acción.

No es necesario que la subroutina de la interrupción 28h chequee el indicador de estado INDOS del DOS, porque como se explicó anteriormente, muchas funciones del DOS del orden más alto que la función 0Ch, pueden ser libremente llamadas desde dentro de una subroutina 28h. Nótese aquí un punto de interés. La subroutina PRINCIPAL puede ser invocada ya sea desde la subroutina TIMER o desde la subroutina BACKPROC. Como hemos visto, desde la subroutina de la interrupción 28h sólo se pueden realizar llamados a las funciones del DOS mayores a la 0Ch, mientras que desde la subroutina TIMER se pueden usar libremente todos los servicios del DOS. Pero, precisamente, ya que PRINCIPAL no sabe cual de las dos subroutinas (TIMER o BACKPROC) le cedió el control, (y esto cada vez puede ser diferente) se han tomado las precauciones de no ejecutar llamados a ninguna función del DOS que se encuentre en el rango de 1 hasta la 0Ch durante toda la ejecución del programa.

La subroutina PRINCIPAL es la parte esencial de TELEX. Cuando esta toma el control, cedido ya sea por TIMER o BACKPROC, la primera tarea que realiza es modificar el contenido de los registros de pila SS y SP para que apunten a una pila interna reservada al final de la memoria residente del programa.

Previamente, el contenido original de estos registros es almacenado en 4 bytes internos denominados REGISTRO_SS y REGISTRO_SP, de manera que al finalizar la sesión de TELEX,

el contenido de estos registros sea restaurado a su valor inicial . Cabe anotar que antes de hacer este cambio de pila, las interrupciones son desactivadas por medio de una instrucción CLI y activadas por la instrucción STI después del cambio. Esto es necesario, porque el ingreso de alguna interrupción en el momento que se están modificando los registros de pila, puede hacer que el microprocesador utilize una pila errónea con resultados impredecibles. Luego, el contenido de los registros de datos, de índice y de segmento son almacenados en la nueva pila para ser recuperados al final de la sesión.

Un llamado a la subroutina IOSET guarda la dirección del Area de transferencia del Disco (o DTA) y, ejecutando un llamado a la función 2Fh de la INT 21h, se crea una nueva DTA localizada en una porción de la memoria residente. La DTA antigua es reservada en cuatro bytes: OLD_DTA_SEGMENT y OLD_DTA_OFFSET. IOSET también modifica los vectores de las interrupciones 1Bh, 23h, y 24h que son las que evitarán el intento de ruptura de la ejecución del programa (CTRL-BREAK), y la salida de mensajes de error enviados por el DOS en caso de detectar errores de disco. Los errores de disco son gobernados diréctamente desde el programa por la subroutina IO_ERROR que trata la INT 24h, indicándole al DOS que ignore cualquier error fatal de disco. Ambas tareas son cruciales para un programa que ejecute accesos al disco y deben ser tomados muy en cuenta si se desea que éste no interfiera con la ejecución de cierta aplicación no residente.

Después, se llama a la subroutina RS232_IO para configurar el puerto de comunicación con la línea de télex, a una velocidad de transmisión de 50 baudios, 5 bits de datos, no paridad y 1.5 bits de parada. Este paso es muy importante y se ejecuta cada vez que la subroutina de TELEX se active. Con esto se tiene la seguridad de que el puerto de comunicación esté operando con los parámetros de transmisión adecuados.¿ Porqué hacer esto ?, porque la ejecución de cualquier otro programa de comunicación puede modificar la configuración del puerto serial usado por TELEX. Si TELEX configura el puerto serial únicamente en la parte de instalación del programa residente, no hay seguridad de que el puerto vaya a funcionar siempre con esa configuración.

PRINCIPAL llama a la subroutina VIDEO_PARAM para guardar todos los parámetros de video que necesariamente son modificados durante la ejecución del programa. Estos son: el modo de video, la página de visualización activa, y el modo, la forma y la posición del cursor. Para obtener el modo y la página de video activos, se realiza un llamado a la función 15h de la INT 10h del BIOS y los parámetros obtenidos en los registros AL y BH son almacenados en los bytes MODO_DE_VIDEO y PAGINA_DE_VIDEO respectivamente. La función 3h de la INT 10h devuelve en los registros CX y DX la forma y la localización del cursor en la pantalla siendo estos almacenados en MODO_DEL_CURSOR y POSICION_DEL_CURSOR.

VIDEO_PARAM chequea la variable ADAPTADOR y desactiva la salida de video, llamando a la subroutina DESACTIVAR_CGA en

caso de que el adaptador de video usado por el sistema sea uno de color estándar CGA. Luego, al llamar a la subroutina GUARDAR_SCR se almacena todo el contenido de la pantalla en una porción que comienza en la etiqueta AREA_PARA_EL_VIDEO y que usa 4000 bytes (que corresponden a 25 lineas x 80 columnas x 2 bytes/caracter) del espacio de memoria reservado por el programa. Esto es necesario puesto que la información de la pantalla es modificada posteriormente al activar cada una de las ventanas creadas por TELEX y debe ser devuelta intacta a la aplicación suspendida.

Para que las ventanas usadas por TELEX puedan ser mostradas en la pantalla, el modo de video debe estar operando en la forma de texto 80 columnas x 25 lineas (modos 2 o 7 para monocromático o modo 3 para color). En caso de no cumplirse esta condición, TELEX cambia el modo de video que esté corriendo, a uno de los modos especificados y usa un byte indicador del cambio llamado MODO_ERRONEO para señalar que el modo de video debe ser restaurado a su forma inicial. Esto sucede cuando un programa es interrumpido por TELEX en el momento que está usando un modo de video gráfico o de alta resolución (en los que no es posible mezclar en la pantalla gráficos con textos) incluso cuando un programa esté usando el modo texto pero con 40 columnas x 25 lineas en el cual las ventanas de TELEX saldrían distorsionadas.

Para el cambio del modo de video se realiza un llamado a la función 0 de la INT 10h indicando en el registro AL el modo deseado (7 o 3). Está función siempre borrará el contenido de

la pantalla, pero no hay inconveniente de perder la información de la misma puesto que previamente esta fue reservada por la subroutina GUARDAR_SCR.

Los últimos pasos que ejecuta VIDEO_PARAM es crear la ventana principal de TELEX que es la del EDITOR. La subroutina ABRIR_VENTANA_EDITOR escribe el par caracter/atributo que definen la ventana en la memoria de video. Esta ventana es formada de 12 lineas por 80 columnas y se la fija en la parte central de la pantalla. Esta subroutina también muestra la fecha y hora actual y, en la última línea de la ventana todas las opciones que pueden ser usadas por TELEX. Si fuera el caso, la salida de video es nuevamente activada y VIDEO_PARAM finaliza su tarea ocultando el cursor.

Ahora, PRINCIPAL fija en 1 el bit 0 de la variable INDICADORES. Este bit indicará que el EDITOR está activado, y por consiguiente toda combinación posterior de ALT SHIFT IZQ deberá ser ignorada por la subroutina TECLADO.

Por último, PRINCIPAL ejecuta un llamado a la subroutina EDITOR desde donde se encuentran disponibles cada una de sus opciones. Un cursor aparece en la parte superior de la ventana, esperando por el ingreso de un texto o comandos del EDITOR.

Bajo circunstancias normales, la interrupción 16h del BIOS, es usada para tomar cualquier tecla presionada; pero, estas no son circunstancias normales. Cada vez que TELEX solicite el ingreso de un caracter desde el teclado, llama a la

subroutina TOMAR_TECLA en lugar de usar el servicio del BIOS. TOMAR_TECLA realiza lo mismo que la interrupción 16h; es más, también la emplea. La diferencia está en que adicionalmente esta subroutina genera la interrupción 28h una y otra vez mientras espera pacientemente por el ingreso de caracteres desde el teclado. ¿ La razón?: Para permitir compatibilidad con otros programas residentes, tales como SIDEKICK.

Para comprender porque se efectúa esto, veamos lo siguiente: Supongamos que TELEX no genere la interrupción 28h y que tratamos de instalarlo junto con el programa SIDEKICK. Si después de instalar ambos programas residentes y, desde la línea de comandos del DOS tratamos de activar SIDEKICK seguido de TELEX, veremos que la secuencia se ejecuta corréctamente; pero si primero activamos TELEX y luego tratamos de activar SIDEKICK, veremos que éste último no responde, siendo su única respuesta una serie de sonidos con los cuales SIDEKICK nos estaría indicando de que no puede activarse. ¿ La razón?.Lo que sucede es que TELEX interrumpió la ejecución 0Ah del DOS (que es la que está corriendo cuando el DOS se encuentra en la línea de comandos), de modo que SIDEKICK al chequear que el indicador de estado del DOS es igual a 1, trata por lo menos de tomar el control a través de la interrupción 28h. Pero, el indicador del DOS no será fijado a cero hasta que TELEX finalice y devuelva el control al DOS y, dado que TELEX no está generando una interrupción 28h, no existe forma de que SIDEKICK pueda activarse. Esta es la razón de incluir esta interrupción 28h dentro de

TOMAR_TECLA.

Entonces, esta situación es remediada haciendo que TELEX genere su propio llamado a la interrupción 28h mientras espera por el ingreso de teclado. De esta manera, un programa tal como SIDEKICK puede activarse en cualquier momento, aún cuando TELEX haya sido invocado desde una interrupción 28h. Esta es una de las pequeñas cosas que hacen de un programa especial debido a su compatibilidad con otros programas residentes.

Durante su ejecución, TELEX.COM utiliza siete ventanas de video las cuales se describen a continuación:

1: La ventana del EDITOR de mensajes que permite elaborar los diferentes mensajes para su correspondiente envío por la línea de télex. Es la primera ventana disponible por el programa y se activa cuando el usuario presiona la secuencia ALT-SHIFT-IZQ, ya sea desde el sistema operativo o desde la ejecución de algún programa de aplicación e incluso desde otro programa residente.

2: Las ventanas del DIRECTORIO y MENU de opciones del directorio que se utilizan para la manipulación de los archivos de disco. Se activan presionando la tecla F1 estando en el modo de edición de mensajes.

3: Una ventana para observar el contenido de los archivos de disco, utilizada por el directorio. Es equivalente a usar el comando interno TYPE del DOS. Una vez seleccionado un

archivo con las diferentes teclas de movimiento del cursor, y presionando la tecla F1 o INTRO, se activa esta ventana para mostrar el contenido de algún archivo seleccionado.

4: La ventana de configuración del programa que permite definir el INDICATIVO del abonado y el directorio de disco donde se almacenarán los mensajes recibidos. Esta ventana también muestra el puerto de comunicación serial utilizado por la interfase de télex (que puede COM1, COM2 o ninguno) y la velocidad de transmisión de la línea (50, 75 o 110 bauds). Estas dos últimas informaciones sirven como referencia y no pueden ser modificadas durante la ejecución del programa. Se activa desde la ventana del EDITOR al presionar la secuencia SHIFT-F1.

5: Al presionar la secuencia CTRL-V se activa una ventana que ingresa al MODO CONVERSACIONAL en el cual todo caracter digitado será inmediátamente enviado por la línea de télex y todo caracter que se reciba por la línea será mostrado en la posición del cursor de la ventana. Se utiliza para realizar diálogos entre abonados.

6: Por último existe la ventana de envío/recepción de mensajes. Esta ventana muestra todo caracter enviado o recibido por la línea de télex, y no requiere secuencia de teclas para activarse. Automáticamente muestra el ingreso de los caracteres al momento de recepción de mensajes, aún cuando el usuario esté ejecutando cualquier otro programa

de aplicación sin que esto interrumpa sus operaciones.

5.7 EL EDITOR DE TEXTOS

El EDITOR usa un espacio reservado de 7800 bytes para almacenar hasta 100 líneas de 78 caracteres de texto. Sirve para preparar, almacenar, ordenar, corregir, editar, transmitir, recibir e imprimir los mensajes. Es un editor de textos sencillo que realiza todas las funciones básicas para crear, modificar, grabar, leer o imprimir un texto de hasta 7800 caracteres. Internamente posee un puntero de memoria usado como cursor del texto, el cual puede ser controlado usando las diferentes teclas de movimiento del cursor y las teclas de desplazamiento del contenido de la pantalla hacia arriba o hacia abajo.

El Editor toma cada tecla presionada y llama a la subroutina CK_MENU para interpretar el ingreso de los caracteres desde el teclado. Esta subroutina chequea si el código del caracter ingresado pertenece a alguna opción del menú del EDITOR o si se está realizando una acción de movimiento del cursor. CK_MENU llama a otras subroutinas dependiendo de la tecla presionada.

Asi, se han definido las siguiente teclas para el control del cursor y las sigueintes funciones del editor:

| : Desplaza el cursor una posición hacia arriba.

| : Desplaza el cursor una posición hacia abajo.

— : Desplaza el cursor una posición hacia la derecha.

← : Desplaza el cursor una posición hacia la izquierda.

PG UP : Desplazamiento de página hacia arriba (11 líneas de texto)

PG DN : Desplazamiento de página hacia abajo (11 líneas de texto)

TAB : Mueve el cursor 8 posiciones hacia la derecha.

BACKSP: Retrocede el cursor y borra el caracter de su nueva posición.

INTRO : Fija el cursor en la posición inicial de la siguiente línea.

HOME : Fija el cursor al inicio de línea.

END : Fija el cursor al final de línea.

CTRL PG UP: Posiciona el cursor al inicio del texto.

CTRL PG DN: Posiciona el cursor al final del texto.

CTRL [vocal][n][m]: Forma las vocales con tildes y las eñes's

INS: Inserta un espacio en blanco a partir de la posición del cursor.

DEL: Borra el caracter de la posición del cursor.

SHIFT F10: Inserta una linea en la posición del cursor.

ALT F10: Borra el contenido de la línea del cursor.

ESC: Salir, ignorar: En cualquier momento, esta tecla anula o

ignora la acción que se esté ejecutando en cualquiera de las ventanas de TELEX. Sirve también para salir del programa residente y retornar al DOS o a la aplicación suspendida.

F1: Presionando esta tecla se ingresa al DIRECTORIO, desde el cual se pueden ejecutar acciones de borrado, renombrado y movimiento de archivos desde un directorio a otro. Inicialmente **F1** muestra el contenido del directorio usado por TELEX, definido en la configuración. En caso de que el directorio no exista, muestra el directorio actual.

F2: Lee un archivo: En la última línea de la ventana del editor muestra el mensaje '**Leer archivo:**_', y espera por el ingreso del nuevo archivo que será almacenado en el espacio del editor para su aumento o modificación. Si el archivo no existe, se muestra el mensaje '**Archivo no encontrado...**'y nuevamente espera por un nuevo nombre de archivo. Este comando puede ignorarse presionando la tecla ESC.

F3: Graba un archivo: El contenido del editor es grabado en el disco como un archivo con un nombre especificado por el usuario. Muestra el mensaje '**Grabar archivo:**_' y espera por el ingreso del nombre del archivo. En caso de que el archivo haya sido anteriormente leído desde el disco, también muestra el nombre del archivo leído, esperando ser grabado el nuevo texto con el mismo nombre

o se da oportunidad de que éste sea modificado. Los mensajes de error: '**Error grabando archivo...**' o '**Espacio insuficiente en el disco...**' serían mostrados cuando se esté tratando de grabar en un disco defectuoso o en un disco lleno. ESC ignora este comando.

F4: Imprime un texto: El contenido del editor es enviado a la impresora conectada en el puerto LPT1. Antes del envio de los caracteres se realiza un chequeo a la impresora. Durante la impresión del texto, puede presionarse la tecla P para hacer una pausa. Luego, al presionar la tecla **INTRO** continúa la impresión. ESC anula en cualquier instante este comando.

F5: Borra el contenido del texto cargado en la memoria. Previamente, para verificación del borrado se muestra el mensaje '¿ **Borrar texto ? (S/N)**'. En caso de que no exista algún texto en la memoria del editor se muestra el mensaje '**No existe texto alguno para grabar, imprimir, borrar o enviar**'.

F6: Tomar línea. Envía un nivel alto a la señales RTS y DTR del puerto serial para indicarle a IETEL el pedido de línea. Con esto, IETEL debe invertir la línea de télex y enviar la fecha y hora actual. Luego, para marcar algún número de abonado, se debe presionar nuevamente esta tecla. Esta acción toma el número del abonado (que debe estar en la primera línea del editor finalizado con el signo '+') y lo envia por la línea de

télex. Si la línea del abonado a llamar está libre, se realiza la comunicación y se recibe nuevamente el número del abonado y su indicativo como señal de verificación para saber si se ha marcado el número correcto. A partir de este momento se encuentra disponible la línea para el correspondiente envío de mensajes. En caso de que la línea del abonado esté ocupada, averiada o fuera de servicio, el equipo recibirá un código de error.

F7: Enviar un texto. Inmediátamente después de presionar esta tecla, todo texto que se encuentre a partir de la segunda línea del editor será enviado por la línea de comunicación. Cada caracter enviado es mostrado por la ventana de comunicación. Esta acción se realiza independientemente de que la línea de télex esté o no invertida.

F8: Parar envío: Interrumpe el envío por la línea de comunicación, pero sin cerrar la llamada.

F9: Cerrar la llamada. Invierte las señales RTS y DTR indicando el final de comunicación. IETEL pone la línea a su estado normal. Otra forma de cortar la comunicación es colocando al final del texto 4 m's (mmmm) o también con una secuencia de puntos (....). Por lo tanto, cada vez que IETEL detecte una de esta secuencia de caracteres, enviará el tiempo que duró la comunicación

con la central y automáticamente cortará la llamada.

F10: Permite mostrar en la parte inferior de la ventana del EDITOR las opciones adicionales de TELEX.

CTRL V: Conversar. Coloca al editor en modo de diálogo durante la comunicación. En este momento , todo lo que sea digitado será automáticamente enviado por la línea de comunicación. Aquí se pueden realizar diálogos entre abonados de la red télex. Durante este modo, cualquiera de los abonados puede anular la comunicación al presionar la tecla F9, o digitando la secuencia de 4 m's o puntos.

CTRL Q: ¿ Quién es Ud?. Uno de los abonados pregunta con quién está en comunicación. El usuario recibe el indicativo del abonado con quien está en comunicación.

CTRL S: Yo soy: El usuario contesta quién es. Se envía su indicativo.

CTRL G: El parlante tanto del equipo del usuario como del abonado en comunicación emite un corto sonido. Sirve para cambio de diálogo. Señal para indicar fin de envío o recepción de caracteres, dando oportunidad para que el otro abonado conteste. Esto es muy útil puesto que mientras un abonado digita, el otro debe esperar por esta señal de cambio, puesto que si ambos tratan de digitar a la vez, recibirán caracteres erróneos. Esto se debe a que la línea opera en el modo half-dúplex.

SHIFT F1: Configuración. Esta secuencia de teclas muestra una ventana en la que se indica toda la configuración usada por TELEX. Esto es, el puerto serie (COM1 o COM2), la velocidad de transmisión (50,75 o 100 bauds) el directorio que contiene los mensajes de télex recibidos y el indicativo del abonado. Aquí se permiten realizar modificaciones del directorio de télex o del indicativo del abonado.

5.8 EL DIRECTORIO

El DIRECTORIO suministra un conjunto de facilidades para la visualización del contenido de los archivos de disco, para el ordenamiento, borrado y movimiento de archivos desde un directorio a otro y para el cambio de directorios desde la aplicación residente.

DIRECTORIO usa la ventaja que proporcionan las teclas de Funciones. Las funciones F1 a F7 se utilizan para ejecutar acciones sobre un archivo de texto seleccionado. Si F1 (o la tecla INTRO) es presionada, el contenido inicial del texto es inmediátamente mostrado en la pantalla. El texto puede ser visualizado línea a línea o por páginas, usando las teclas de movimiento del cursor. Ctrl-PgUp o HOME muestran el inicio del archivo, mientras que Ctrl-PgDn o END muestran su contenido final.

Mientras se esté visualizando el contenido del archivo, es posible retornar al menú principal del directorio, presionando cualquiera de las teclas: F1, INTRO o ESC. Asi

como la mayoría de las aplicaciones usan la tecla ESC para volver a un paso previo, DIRECTORIO también define esta tecla para realizar tal función. En este caso, ESC permite retornar a las ventanas de DIRECTORIO, del EDITOR, anular algún procedimiento o simplemente retornar hacia el Sistema Operativo o programa de aplicación que fue interceptado por TELEX.

La tecla de función F2 sirve para Editar un archivo, es decir, permite leer el contenido del archivo hacia el espacio de memoria reservado para el editor de textos, para que sea modificado, enviado a la impresora o por el puerto de comunicación. Antes de leer el archivo, DIRECTORIO muestra un mensaje de verificación, puesto que puede existir en el EDITOR algún texto que no haya sido grabado al disco. Cabe anotar que si el número de bytes del contenido del archivo es mayor que el número de bytes reservado para el EDITOR, el archivo será parcialmente cargado a la memoria.

La tecla F3 permite renombrar un archivo y la tecla F4 mover el archivo desde un directorio a otro directorio. Cuando F3 es presionada, el cursor se activa bajo la ventana del menú del directorio y espera por el ingreso de un nuevo nombre para el archivo seleccionado. El nuevo nombre para el archivo no debe contener los caracteres : \ ? * . Una vez ingresado el nuevo nombre del archivo, y presionando la tecla INTRO, el nuevo nombre del archivo remplazará al anterior tanto en el disco como en el listado del directorio. Si se ingresó un caracter inválido tal como un espacio, o uno de

los caracteres mencionados anteriormente, el directorio responderá con un sonido indicando el error. Al presionar la tecla **ESC** se ignora esta acción de renombrar un archivo.

La tecla **F4** mueve un archivo de un directorio a otro Al igual que **F3**, cuando se presiona **F4**, aparece el cursor para el ingreso de un nuevo directorio para el archivo. DIRECTORIO no permite mover un archivo de una unidad de disco a otra, ni tampoco permite moverlo a un directorio que ya tiene un archivo con el mismo nombre del archivo seleccionado. Si se desea mover un archivo a otro directorio y también darle un nuevo nombre, primero podemos presionar **F3**, renombrar el archivo y luego usar **F4** para moverlo al nuevo directorio. Asi como existe un formato de confirmación para borrar un archivo, también existe uno para moverlo a otro directorio.

En ciertas ocasiones de desea borrar algún archivo o texto. la tecla **F5** ha sido destinada para realizar esa acción. Como una precaución, antes de borrar el archivo seleccionado, se espera el ingreso de la tecla 'S' para confirmar el borrado del archivo, o la tecla 'N' o **ESC** para cancelar la acción de borrar archivo. Si no se desea confirmar el borrado de archivos del disco, la tecla **F6** elimina el paso de confirmación. **F6** es un conmutador; su valor inicial de confirmación es 'SI'. Si el modo de confirmación es 'NO' y se presiona F5, el archivo será borrado automáticamente del disco. Después de borrar un archivo, el directorio se actualiza eliminando de su listado el nombre del archivo y

disminuye en 1 el contador del número de archivos que contiene el directorio.

Con la tecla de función F7 existe la opción de cambio de directorio. El cursor es activado para permitir el ingreso del nuevo directorio por mostrar. Si el nombre del directorio ingresado no existe se muestra el siguiente mensaje de error '**Directorio erróneo**', y el comando de cambio directorio es ignorado.

Las últimas dos funciones del directorio controlan las diferentes formas de visualizar el orden del listado de los archivos del directorio. Estas funciones son **F9**, ordenar por nombres de archivo y **F10**, ordenar por fecha de creación del archivo.

Cabe anotar que las funciones para borrar, renombrar y mover archivos, trabajan diréctamente mientras se esté observando un archivo. Por ejemplo, si después de presionar la tecla **F1** para visualizar el contenido de un archivo, decidimos borrarlo, en lugar de retornar primero al menú con **ESC**, simplemente podemos presionar **F5** e inmediátamente el programa retornará al punto de confirmación de borrado del archivo (asumiendo que el indicador de confirmación esté puesto en '**SI**', caso contrario, el archivo es borrado inmediátamente).

Si listamos el directorio raiz de un disco conteniendo el arranque del Sistema Operativo, encontraremos los dos archivos ocultos del sistema del DOS, **IBMBIO.COM** e

IBMDOS.COM, incluidos en el listado. Estos archivos son omitidos por el comando interno **DIR** del DOS. DIRECTORIO lista cualquier archivo oculto o del sistema que encuentre. Para distinguir los archivos ocultos de los archivos normales, DIRECTORIO coloca una 'H' inmediátamente después de la fecha de creación del archivo. Los archivos ocultos o del sistema pueden ser libremente observados usando **F1**, pero no pueden ser borrados, renombrados o movidos de directorio. Si se intenta ejecutar una de tales acciones, el programa simplemente responde con un sonido.

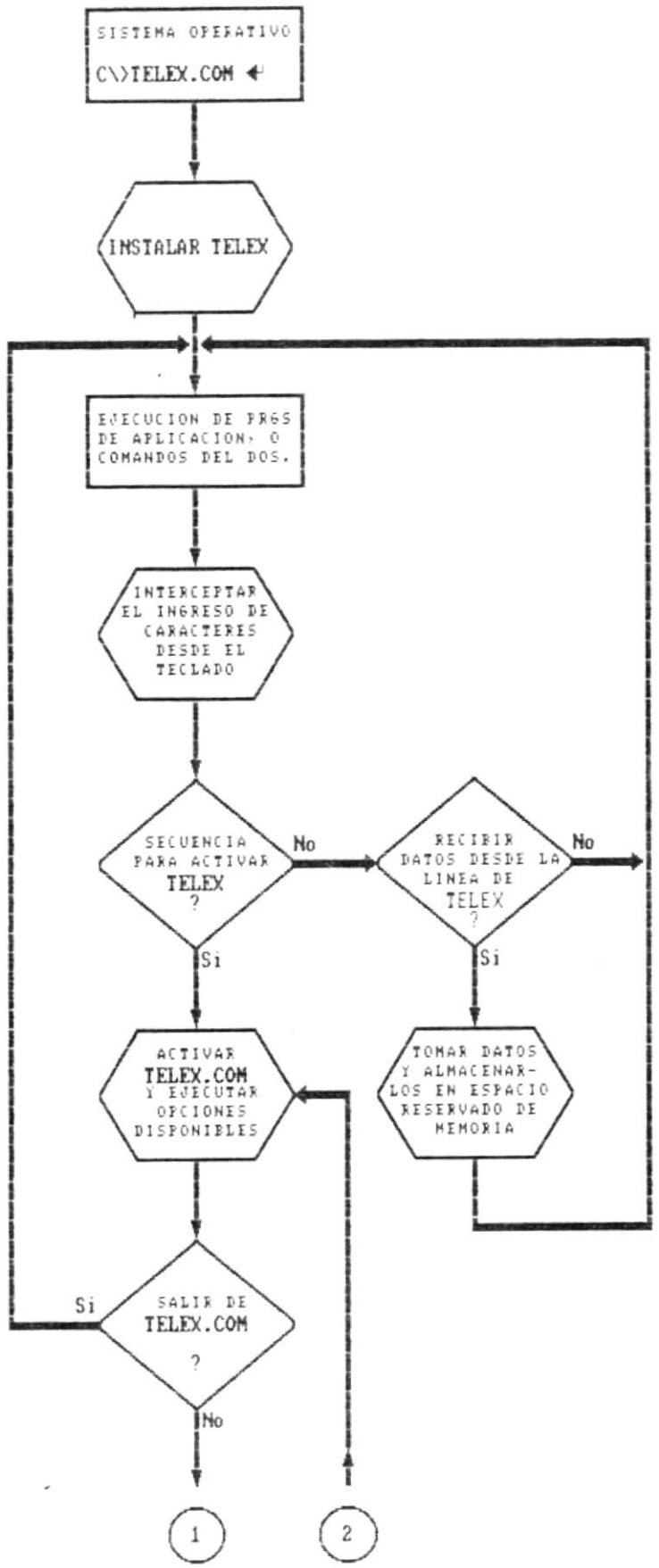
Cuando es presionada la tecla ESC cerrar la ventana del DIRECTORIO y luego para cerrar la ventana del EDITOR, algunos cambios ocurren internamente:

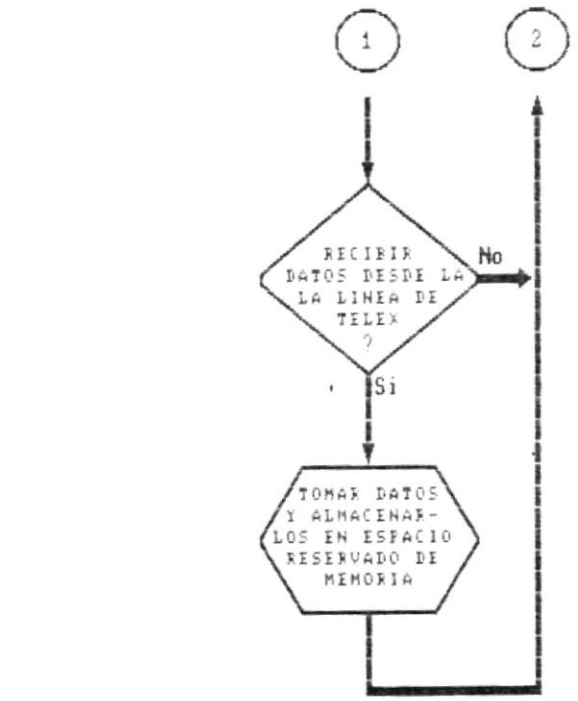
1: La dirección del Area de Transferencia del Disco (o DTA) original y los vectores de interrupción de 1Bh, 23H y 24h son restaurados a sus valores iniciales.

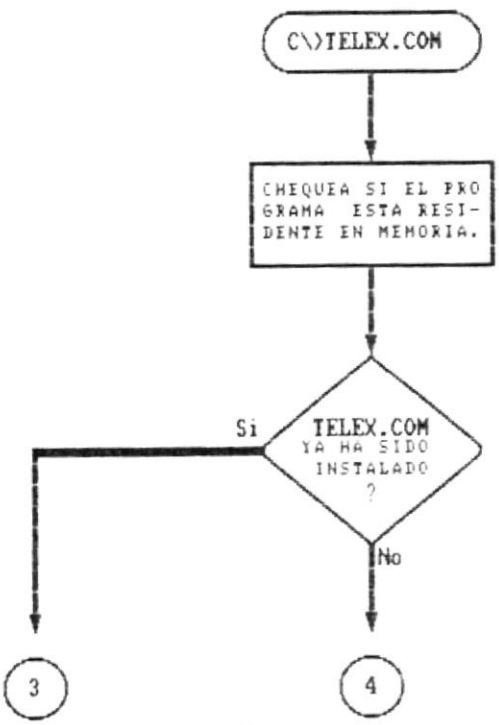
2: El contenido de la pantalla que fue guardado antes de abrir la ventana del Editor, es restaurado a la memoria de video.

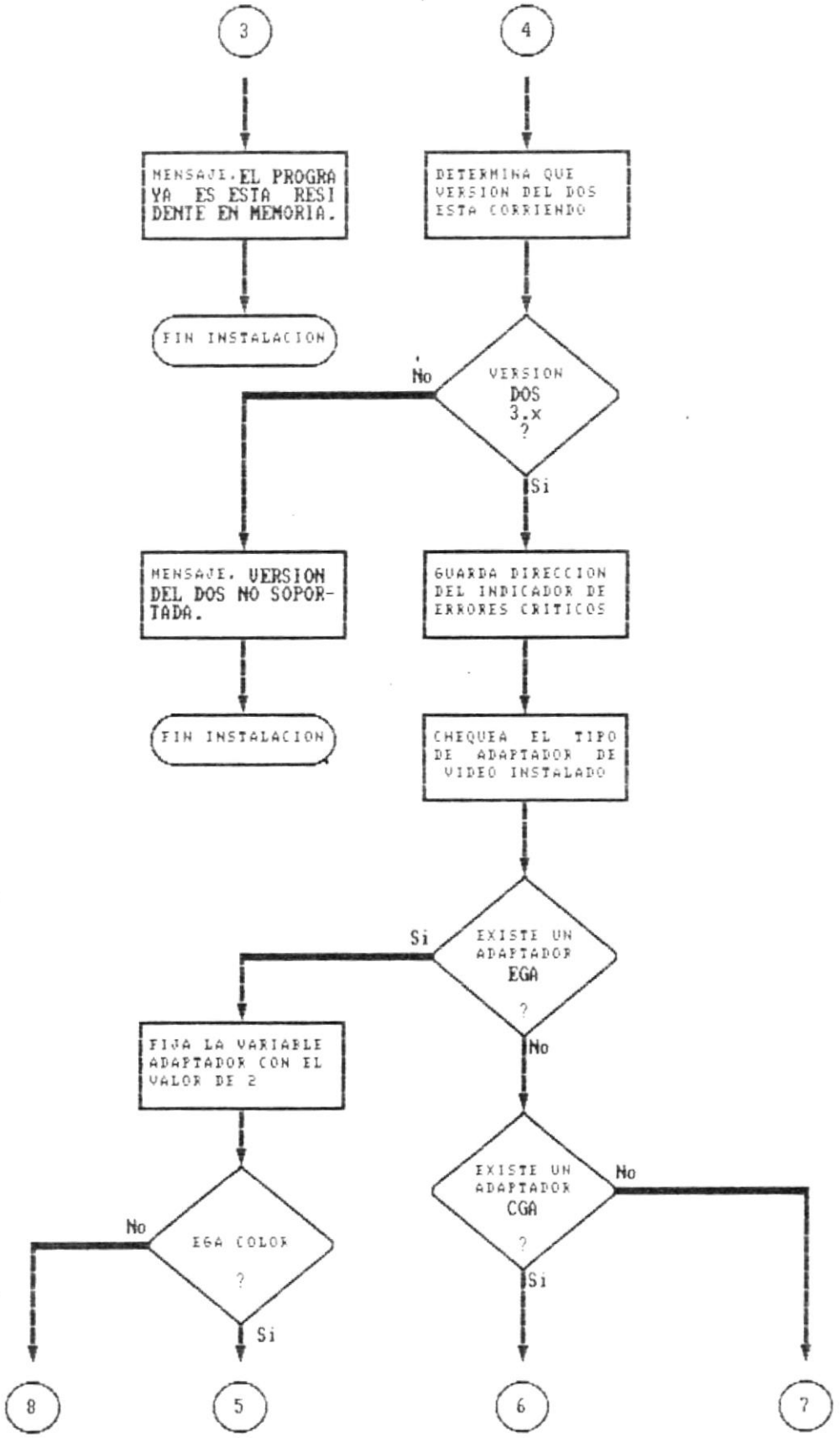
3: Finalmente, el cursor es fijado a su forma y localización original y la subroutina PRINCIPAL finaliza retornando al manipulador de interrupciones que la invocó, ya sea TIMER o BACKPROC. El manipulador (en turno) transfiere o devuelve el control al programa interrumpido por la combinación ALT_SHIFT_IZQ.

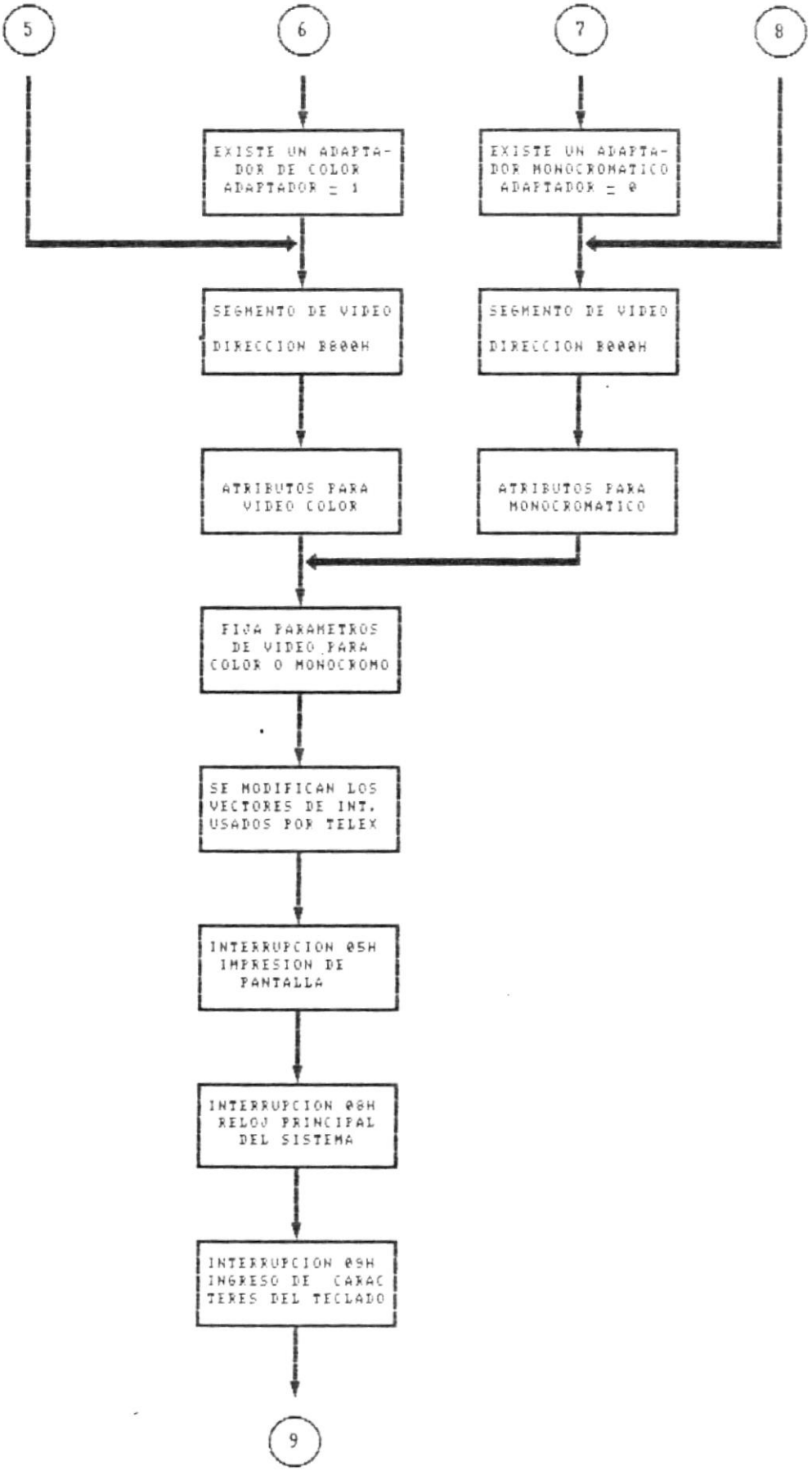
El programa global del sistema de control está constituido por un lazo principal y una serie de subrutinas cuyos diagramas de flujo y listado se exponen a continuación.

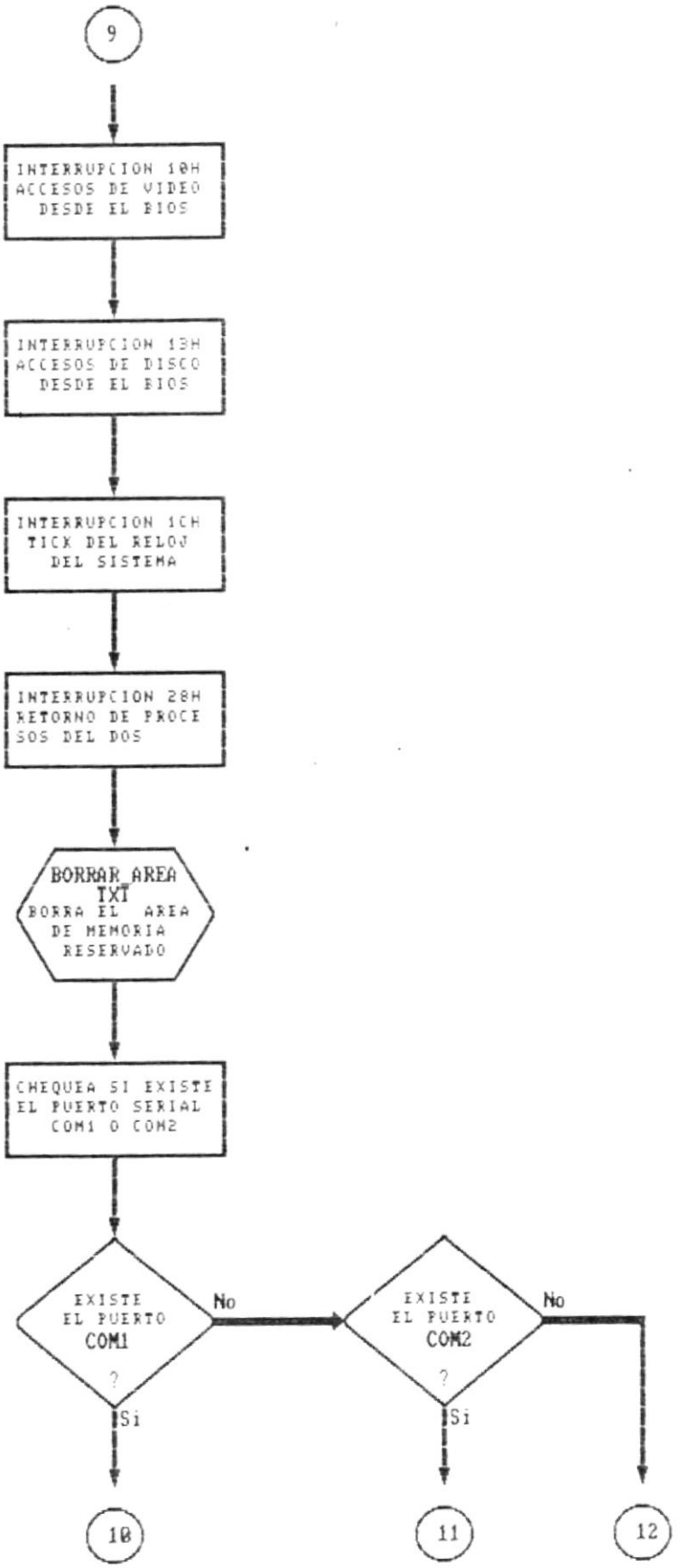

SISTEMA OPERATIVO
C\>TELEX.COM ←
INSTALAR TELEX
EJECUCION DE PRGS DE APLICACION, O COMANDOS DEL DOS.
INTERCEPTAR EL INGRESO DE CARACTERES DESDE EL TECLADO
SECUENCIA PARA ACTIVAR TELEX ?
No
Si
RECIBIR DATOS DESDE LA LINEA DE TELEX ?
No
Si
ACTIVAR TELEX.COM Y EJECUTAR OPCIONES DISPONIBLES
TOMAR DATOS Y ALMACENAR-LOS EN ESPACIO RESERVADO DE MEMORIA
SALIR DE TELEX.COM ?
Si
No
1
2

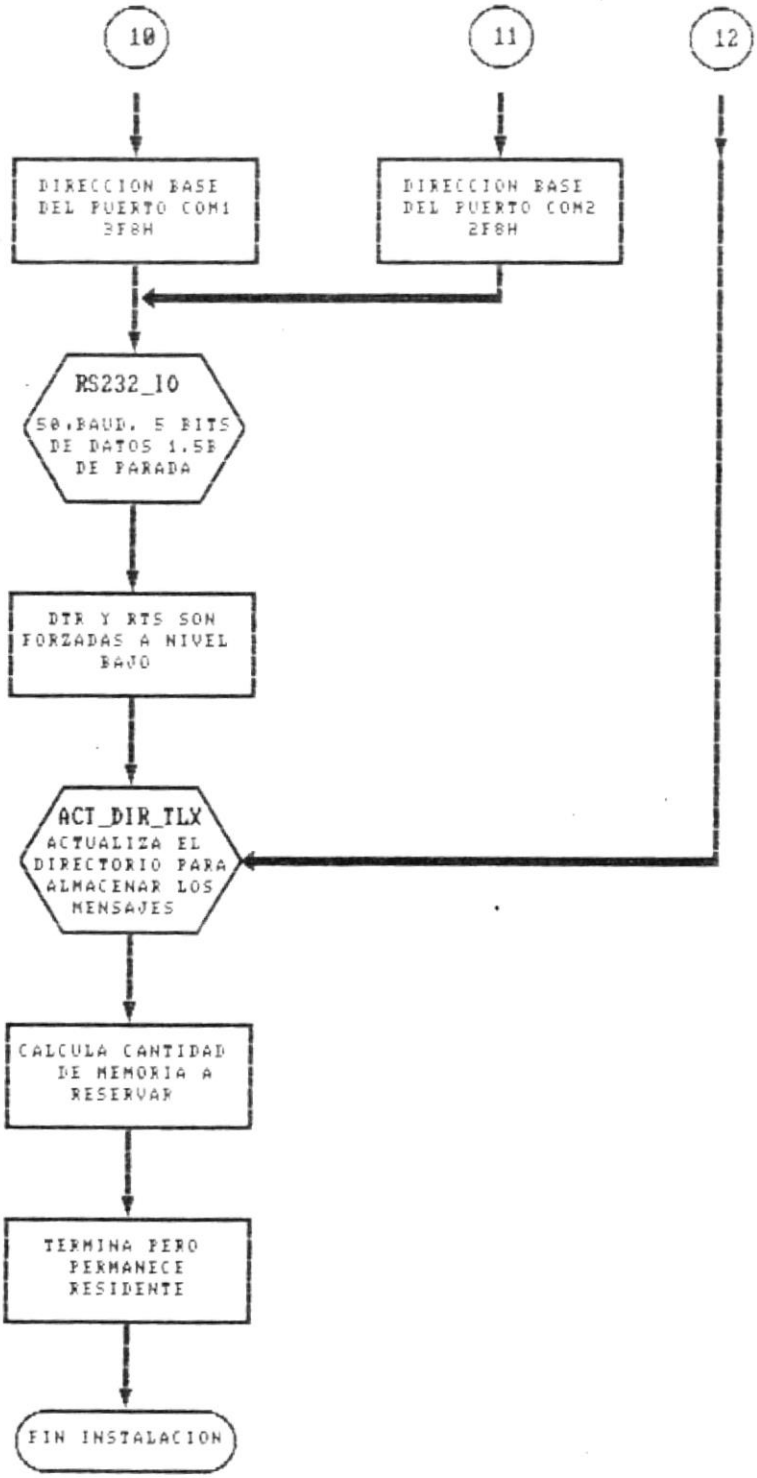

1
2
RECIBIR DATOS DESDE LA LA LINEA DE TELEX ?
No
Si
TOMAR DATOS Y ALMACENAR-LOS EN ESPACIO RESERVADO DE MEMORIA

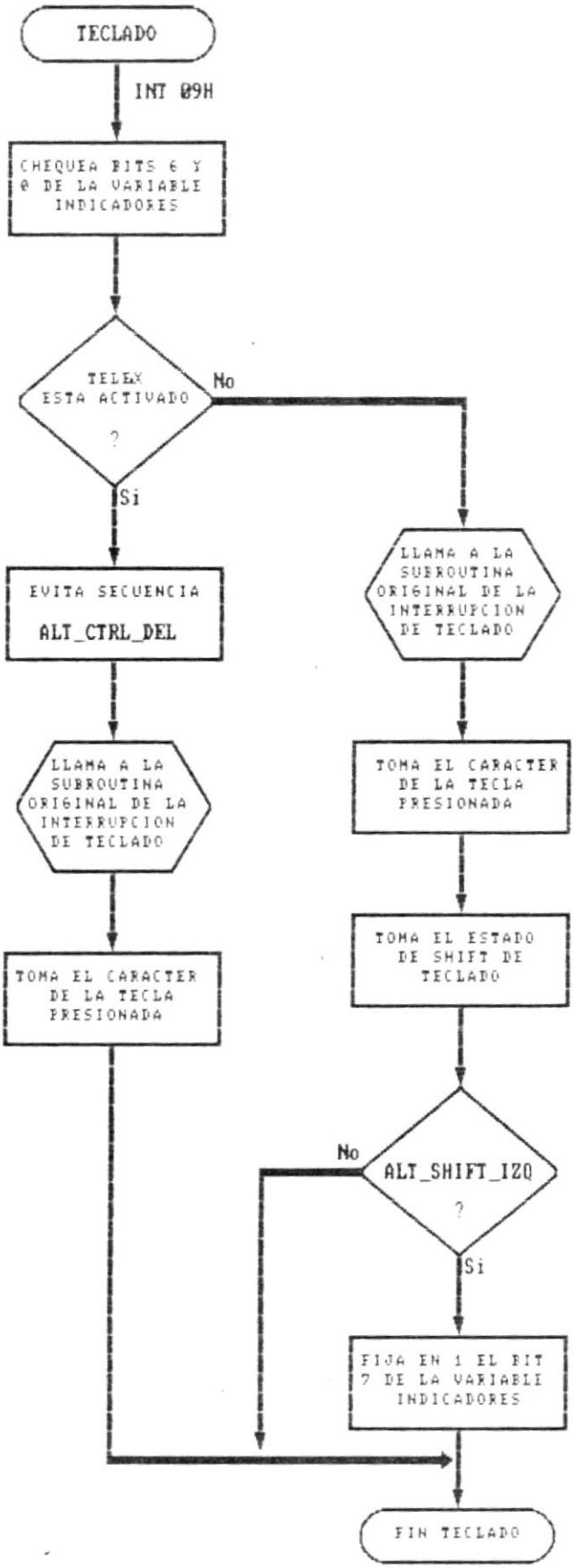

C\)TELEX.COM
CHEQUEA SI EL PRO GRAMA ESTA RESI-DENTE EN MEMORIA.
Si
TELEX.COM YA HA SIDO INSTALADO ?
No
3
4

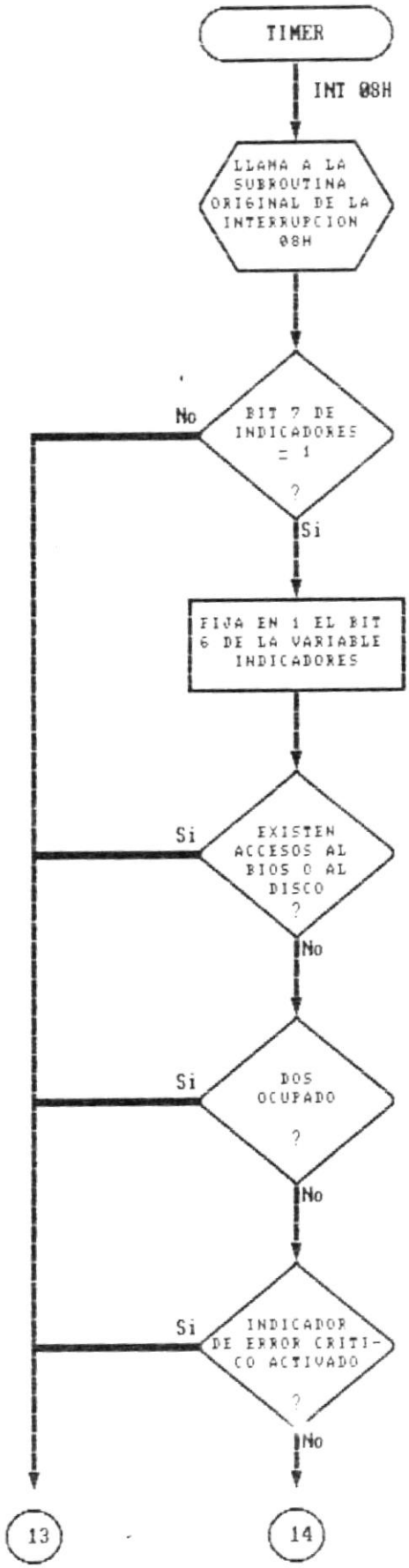

3
4
MENSAJE. EL PROGRA YA ES ESTA RESI DENTE EN MEMORIA.
DETERMINA QUE VERSION DEL DOS ESTA CORRIENDO
FIN INSTALACION
No
VERSION DOS 3.x ?
Si
MENSAJE. VERSION DEL DOS NO SOPOR-TADA.
GUARDA DIRECCION DEL INDICADOR DE ERRORES CRITICOS
FIN INSTALACION
CHEQUEA EL TIPO DE ADAPTADOR DE VIDEO INSTALADO
Si
EXISTE UN ADAPTADOR EGA ?
No
FIJA LA VARIABLE ADAPTADOR CON EL VALOR DE 2
No
EGA COLOR ?
Si
EXISTE UN ADAPTADOR CGA ?
No
Si
8
5
6
7

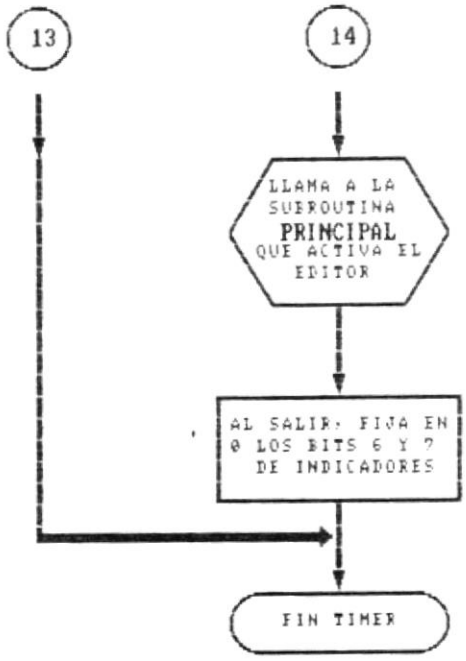

5
6
7
8
EXISTE UN ADAPTADOR DE COLOR ADAPTADOR = 1
EXISTE UN ADAPTADOR MONOCROMATICO ADAPTADOR = 0
SEGMENTO DE VIDEO DIRECCION B800H
SEGMENTO DE VIDEO DIRECCION B000H
ATRIBUTOS PARA VIDEO COLOR
ATRIBUTOS PARA MONOCROMATICO
FIJA PARAMETROS DE VIDEO PARA COLOR O MONOCROMO
SE MODIFICAN LOS VECTORES DE INT. USADOS POR TELEX
INTERRUPCION 05H IMPRESION DE PANTALLA
INTERRUPCION 08H RELOJ PRINCIPAL DEL SISTEMA
INTERRUPCION 09H INGRESO DE CARACTERES DEL TECLADO
9

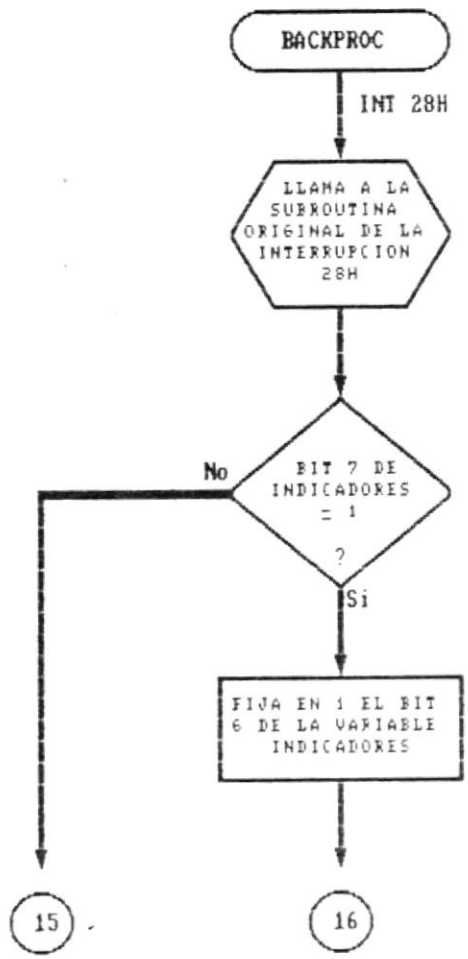

9
INTERRUPCION 10H
ACCESOS DE VIDEO
DESDE EL BIOS
INTERRUPCION 13H
ACCESOS DE DISCO
DESDE EL BIOS
INTERRUPCION 1CH
TICK DEL RELOJ
DEL SISTEMA
INTERRUPCION 28H
RETORNO DE PROCE
SOS DEL DOS
BORRAR AREA
TXT
BORRA EL AREA
DE MEMORIA
RESERVADO
CHEQUEA SI EXISTE
EL PUERTO SERIAL
COM1 O COM2
EXISTE
EL PUERTO
COM1
?
No
Si
EXISTE
EL PUERTO
COM2
?
No
Si
10
11
12

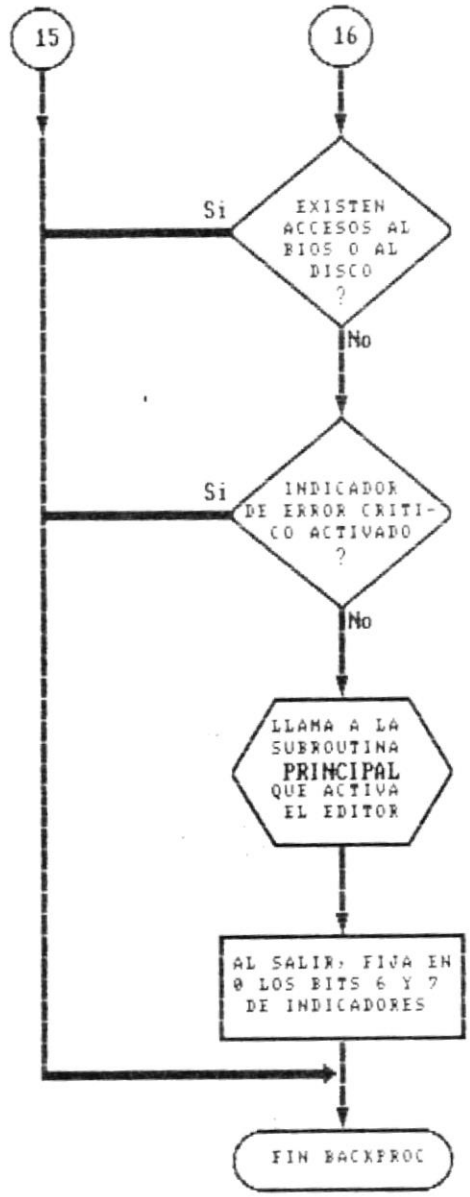

10
11
12
DIRECCION BASE DEL PUERTO COM1 3F8H
DIRECCION BASE DEL PUERTO COM2 2F8H
RS232_IO
50.BAUD. 5 BITS DE DATOS 1.5B DE PARADA
DTR Y RTS SON FORZADAS A NIVEL BAJO
ACT_DIR_TLX
ACTUALIZA EL DIRECTORIO PARA ALMACENAR LOS MENSAJES
CALCULA CANTIDAD DE MEMORIA A RESERVAR
TERMINA PERO PERMANECE RESIDENTE
FIN INSTALACION


TECLADO
INT 09H
CHEQUEA BITS 6 Y 0 DE LA VARIABLE INDICADORES
TELEX ESTA ACTIVADO ?
No
Si
EVITA SECUENCIA ALT_CTRL_DEL
LLAMA A LA SUBROUTINA ORIGINAL DE LA INTERRUPCION DE TECLADO
TOMA EL CARACTER DE LA TECLA PRESIONADA
LLAMA A LA SUBROUTINA ORIGINAL DE LA INTERRUPCION DE TECLADO
TOMA EL CARACTER DE LA TECLA PRESIONADA
TOMA EL ESTADO DE SHIFT DE TECLADO
No
ALT_SHIFT_IZQ ?
Si
FIJA EN 1 EL BIT 7 DE LA VARIABLE INDICADORES
FIN TECLADO

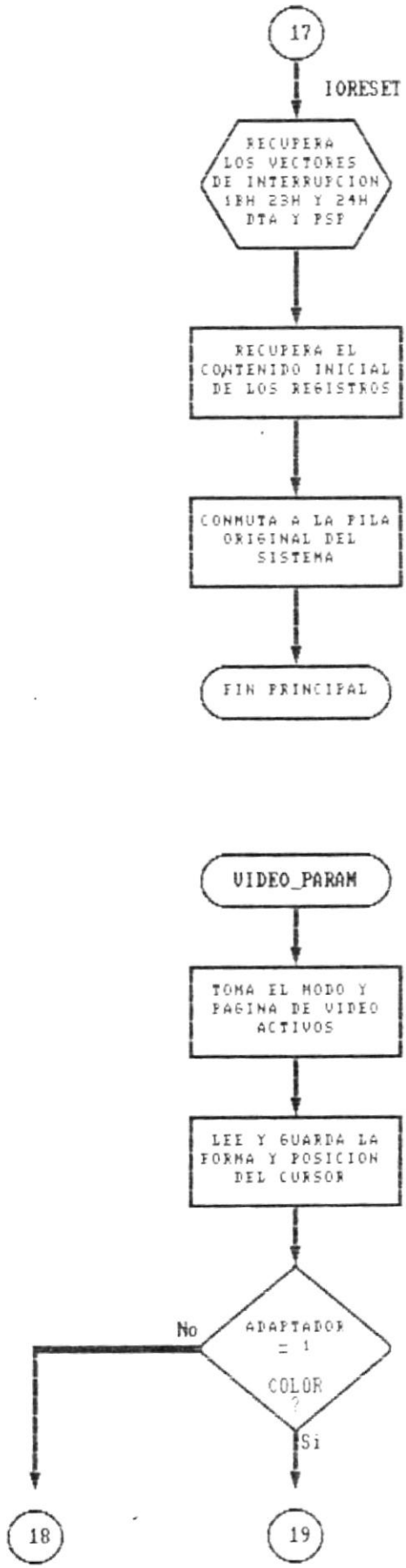

TIMER
INT 08H
LLAMA A LA SUBROUTINA ORIGINAL DE LA INTERRUPCION 08H
BIT 7 DE INDICADORES = 1 ?
No
Si
FIJA EN 1 EL BIT 6 DE LA VARIABLE INDICADORES
EXISTEN ACCESOS AL BIOS O AL DISCO ?
Si
No
DOS OCUPADO ?
Si
No
INDICADOR DE ERROR CRITICO ACTIVADO ?
Si
No
13
14

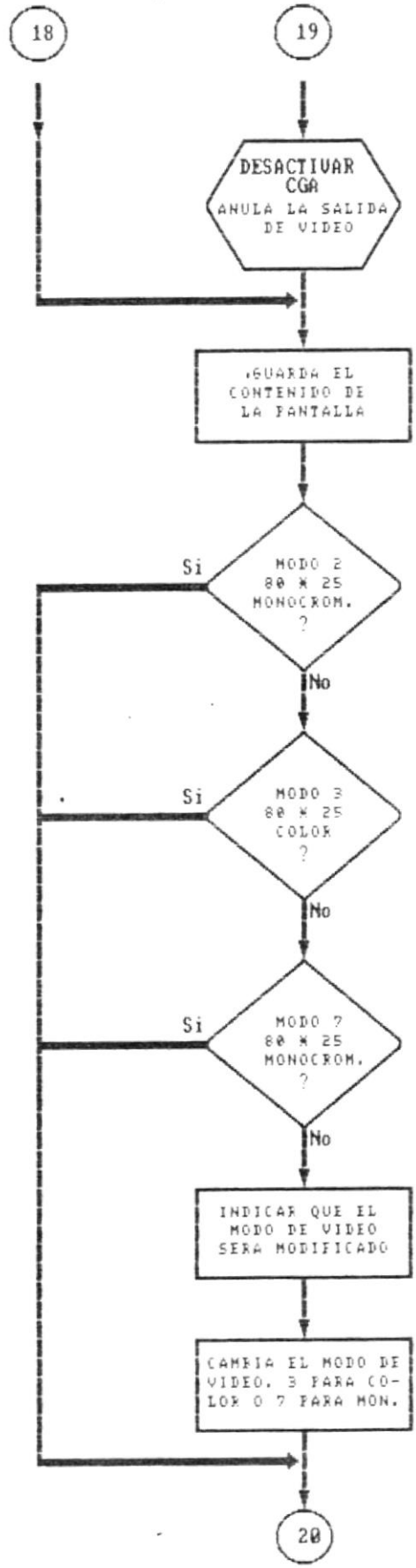

13
14
LLAMA A LA SUBROUTINA PRINCIPAL QUE ACTIVA EL EDITOR
AL SALIR: FIJA EN 0 LOS BITS 6 Y 7 DE INDICADORES
FIN TIMER

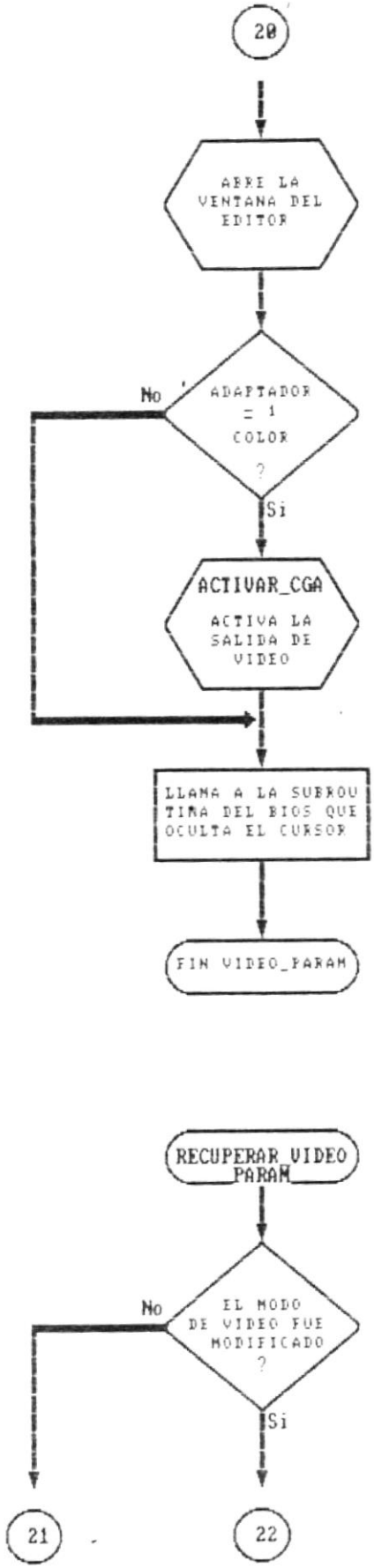

BACKPROC
INT 28H
LLAMA A LA SUBROUTINA ORIGINAL DE LA INTERRUPCION 28H
No
BIT 7 DE INDICADORES = 1 ?
Si
FIJA EN 1 EL BIT 6 DE LA VARIABLE INDICADORES
15
16

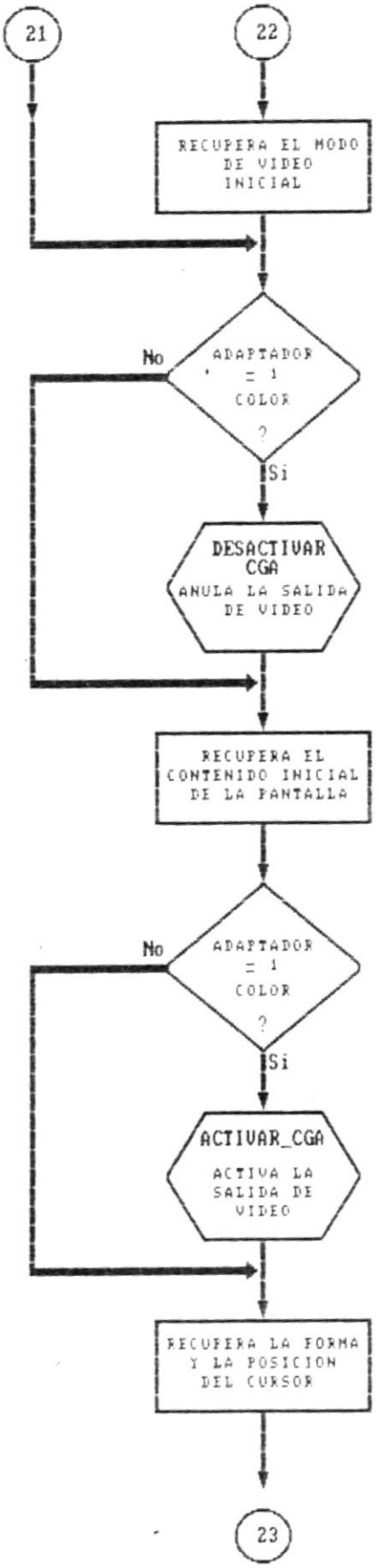

15
16
EXISTEN ACCESOS AL BIOS O AL DISCO ?
Si
No
INDICADOR DE ERROR CRITICO ACTIVADO ?
Si
No
LLAMA A LA SUBROUTINA PRINCIPAL QUE ACTIVA EL EDITOR
AL SALIR: FIJA EN 0 LOS BITS 6 Y 7 DE INDICADORES
FIN BACKPROC

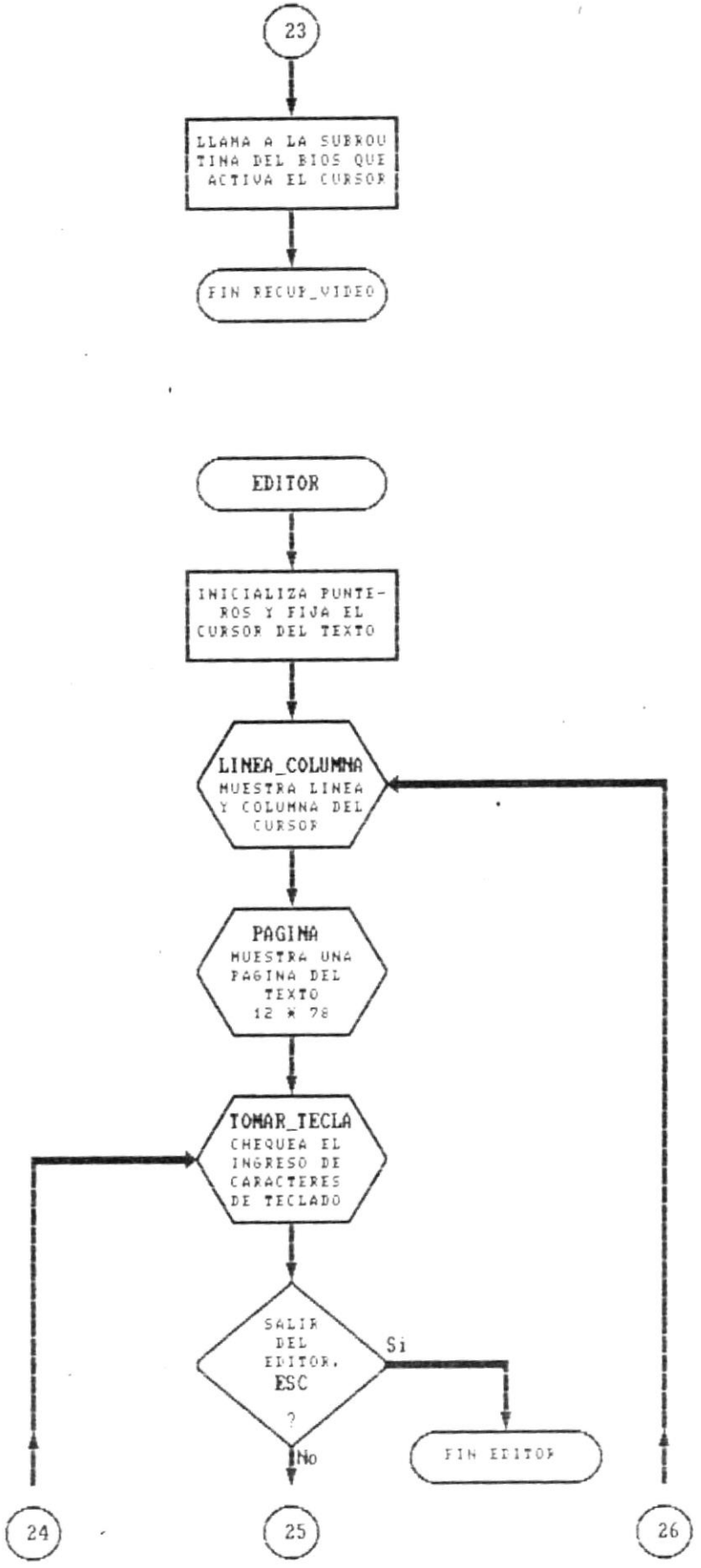

PRINCIPAL
ACTIVA LA PILA INTERNA DE TELEX Y GUARDA LOS REG.
IOSET
MODIFICA LOS VECTORES DE LAS INT'S 1BH 23H Y 24H DTA Y PSP
RS232_IO
CONFIGURA EL PUERTO SERIAL COM1 O COM2
VIDEO_PARAM
CHEQUEA LOS PARAMETROS DE VIDEO
INDICADORES
BIT 0 = 1
EDITOR
REALIZA TODAS FUNCIONES DISPONIBLES DE TELEX
BIT 0 DE LA VARIABLE INDICADORES = 0
RECUPERAR VIDEO_PARAM
RECUPERA LOS PARAMETROS DE VIDEO
17

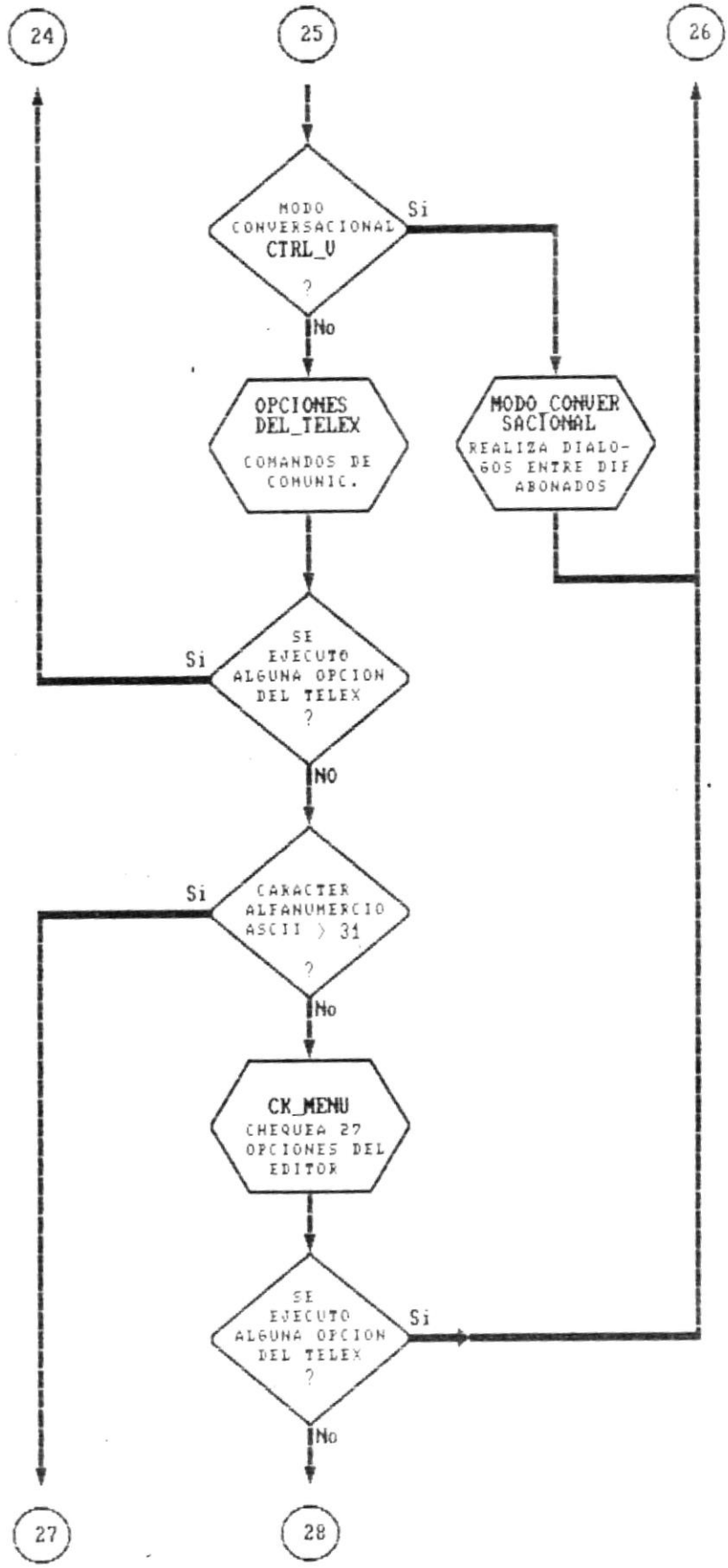

17
IORESET
RECUPERA LOS VECTORES DE INTERRUPCION 1BH 23H Y 24H DTA Y PSP
RECUPERA EL CONTENIDO INICIAL DE LOS REGISTROS
CONMUTA A LA PILA ORIGINAL DEL SISTEMA
FIN PRINCIPAL
VIDEO_PARAM
TOMA EL MODO Y PAGINA DE VIDEO ACTIVOS
LEE Y GUARDA LA FORMA Y POSICION DEL CURSOR
ADAPTADOR = 1
COLOR ?
No
Si
18
19

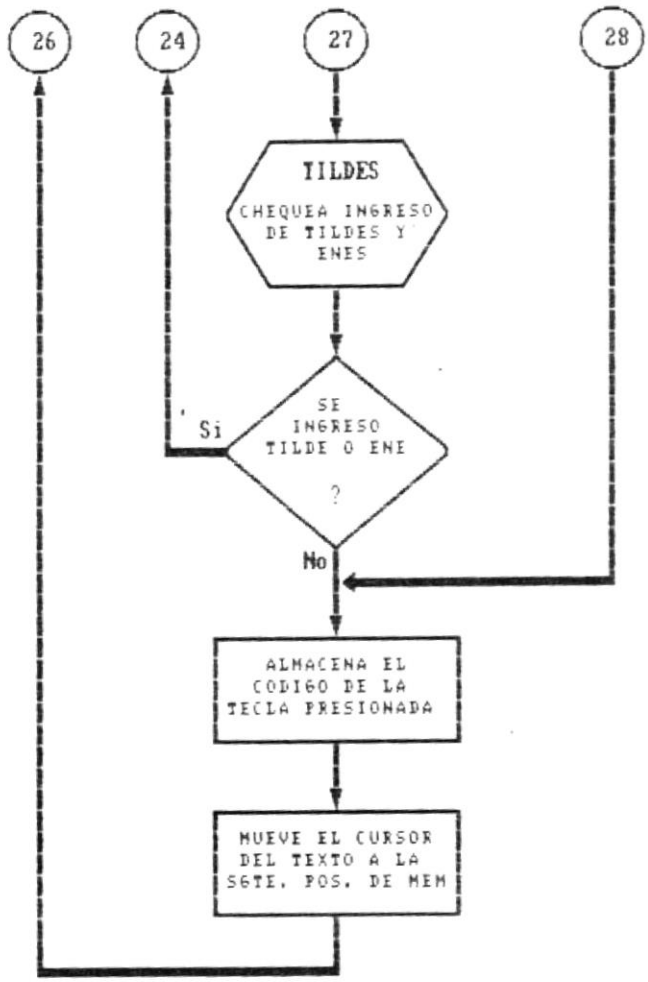

18
19
DESACTIVAR CGA
ANULA LA SALIDA DE VIDEO
¡GUARDA EL CONTENIDO DE LA PANTALLA
MODO 2 80 X 25 MONOCROM. ?
Si
No
MODO 3 80 X 25 COLOR ?
Si
No
MODO 7 80 X 25 MONOCROM. ?
Si
No
INDICAR QUE EL MODO DE VIDEO SERA MODIFICADO
CAMBIA EL MODO DE VIDEO. 3 PARA COLOR O 7 PARA MON.
20

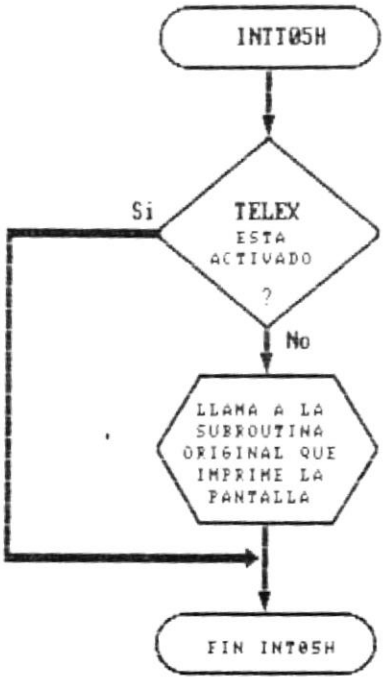

20
ABRE LA VENTANA DEL EDITOR
ADAPTADOR = 1 COLOR ?
No
Si
ACTIVAR_CGA
ACTIVA LA SALIDA DE VIDEO
LLAMA A LA SUBROUTINA DEL BIOS QUE OCULTA EL CURSOR
FIN VIDEO_PARAM
RECUPERAR_VIDEO PARAM
EL MODO DE VIDEO FUE MODIFICADO ?
No
Si
21
22


21
22
RECUPERA EL MODO DE VIDEO INICIAL
ADAPTADOR = 1 COLOR ?
No
Si
DESACTIVAR CGA
ANULA LA SALIDA DE VIDEO
RECUPERA EL CONTENIDO INICIAL DE LA PANTALLA
ADAPTADOR = 1 COLOR ?
No
Si
ACTIVAR_CGA
ACTIVA LA SALIDA DE VIDEO
RECUPERA LA FORMA Y LA POSICION DEL CURSOR
23

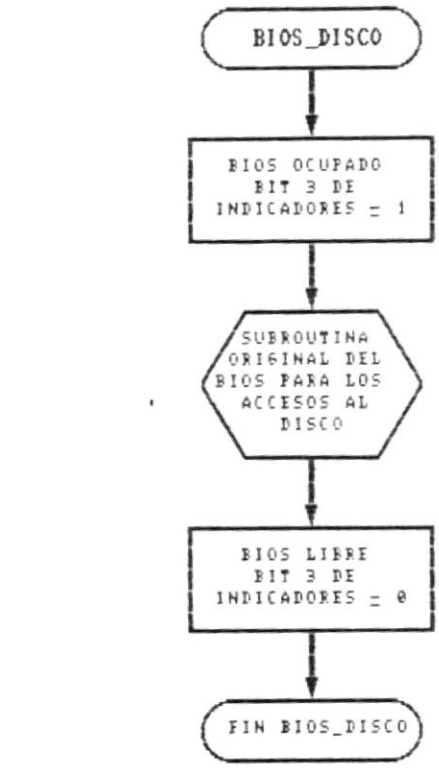

23
LLAMA A LA SUBROU TINA DEL BIOS QUE ACTIVA EL CURSOR
FIN RECUP_VIDEO
EDITOR
INICIALIZA PUNTE- ROS Y FIJA EL CURSOR DEL TEXTO
LINEA_COLUMNA
MUESTRA LINEA Y COLUMNA DEL CURSOR
PAGINA
MUESTRA UNA PAGINA DEL TEXTO 12 * 78
TOMAR_TECLA
CHEQUEA EL INGRESO DE CARACTERES DE TECLADO
SALIR DEL EDITOR, ESC ?
Si
No
FIN EDITOR
24
25
26

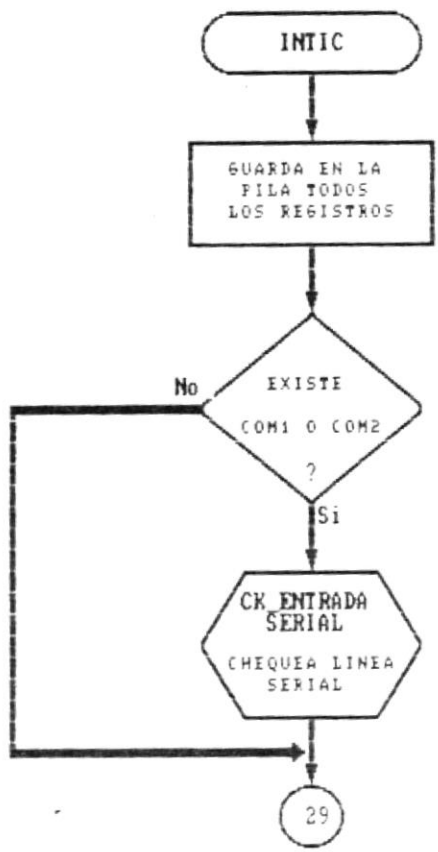

24
25
26
MODO CONVERSACIONAL CTRL_U ?
Si
No
OPCIONES DEL_TELEX
COMANDOS DE COMUNIC.
MODO_CONVER SACIONAL
REALIZA DIALO-GOS ENTRE DIF ABONADOS
SE EJECUTO ALGUNA OPCION DEL TELEX ?
Si
NO
CARACTER ALFANUMERCIO ASCII > 31 ?
Si
No
CK_MENU
CHEQUEA 27 OPCIONES DEL EDITOR
SE EJECUTO ALGUNA OPCION DEL TELEX ?
Si
No
27
28

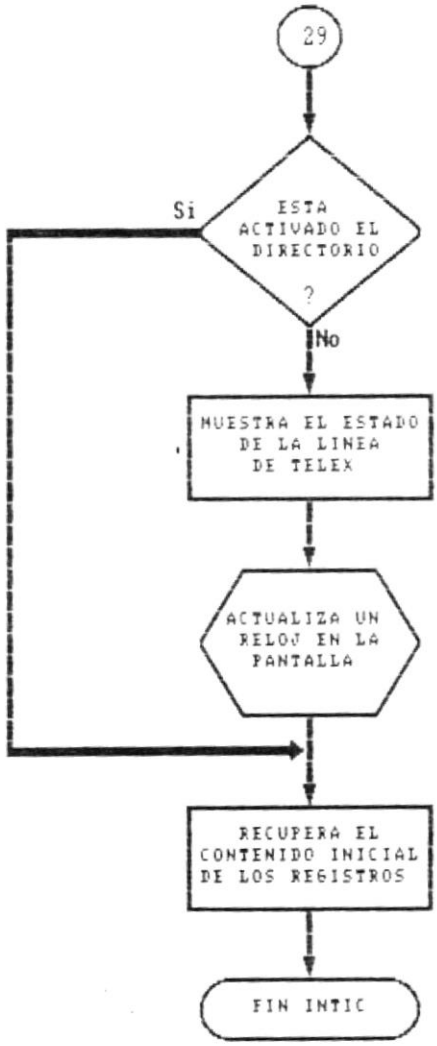

26
24
27
28
TILDES
CHEQUEA INGRESO DE TILDES Y ENES
SE INGRESO TILDE O ENE ?
Si
No
ALMACENA EL CODIGO DE LA TECLA PRESIONADA
MUEVE EL CURSOR DEL TEXTO A LA SGTE. POS. DE MEM


INTØ5H
TELEX ESTA ACTIVADO ?
Si
No
LLAMA A LA SUBROUTINA ORIGINAL QUE IMPRIME LA PANTALLA
FIN INTØ5H

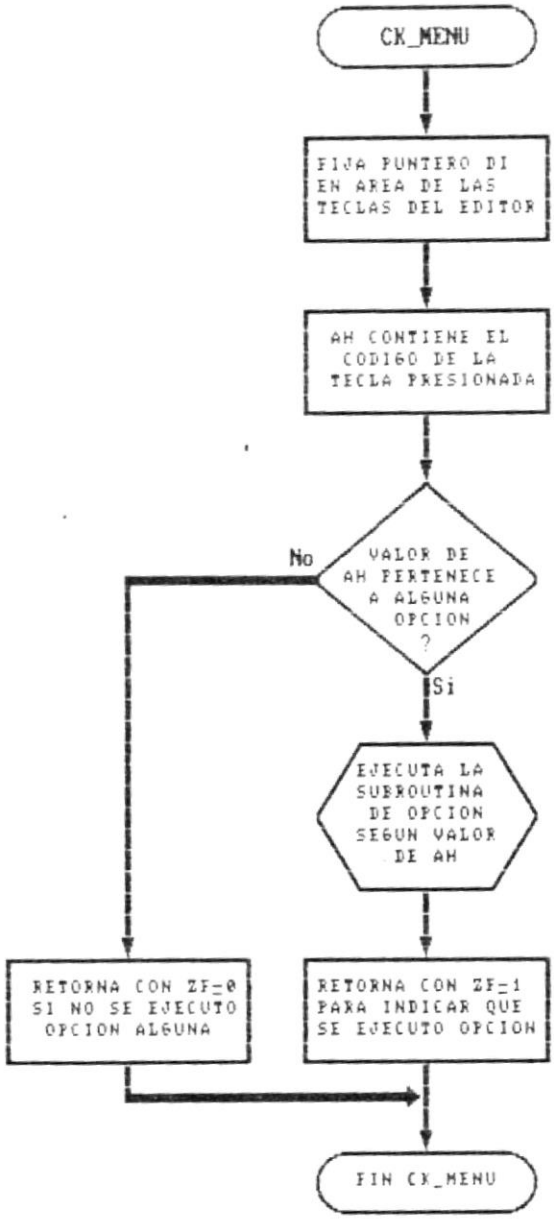

VIDEO
INDICAR QUE EL BIOS ESTA OCUPADO B2 INDICADORES=1
LLAMA A LA RUTINA ORIGINAL DEL BIOS
BIOS LIBRE
B2 INDICADORES=Ø
FIN VIDEO


BIOS_DISCO
BIOS OCUPADO
BIT 3 DE
INDICADORES = 1
SUBROUTINA
ORIGINAL DEL
BIOS PARA LOS
ACCESOS AL
DISCO
BIOS LIBRE
BIT 3 DE
INDICADORES = 0
FIN BIOS_DISCO


INTIC
GUARDA EN LA
PILA TODOS
LOS REGISTROS
No
EXISTE
COM1 O COM2
?
Si
CK_ENTRADA
SERIAL
CHEQUEA LINEA
SERIAL
29

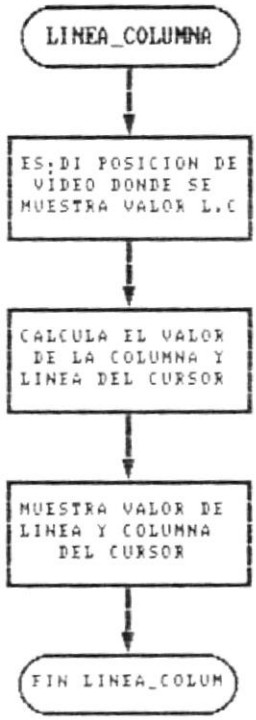

29
ESTA ACTIVADO EL DIRECTORIO ?
Si
No
MUESTRA EL ESTADO DE LA LINEA DE TELEX
ACTUALIZA UN RELOJ EN LA PANTALLA
RECUPERA EL CONTENIDO INICIAL DE LOS REGISTROS
FIN INTIC

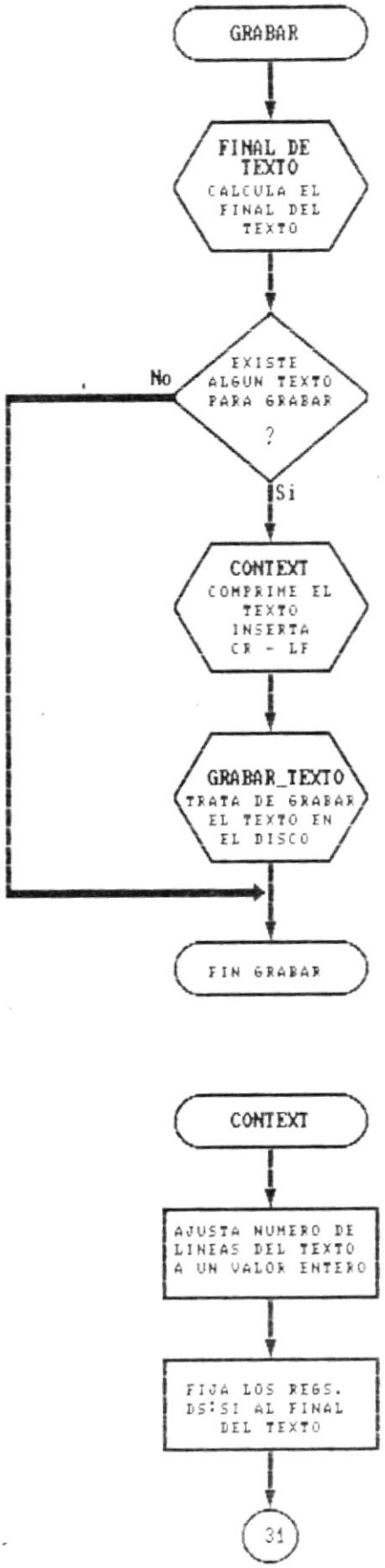

IO_ERROR
INDICA AL DOS QUE IGNORE CUALQUIER TIPO DE ERROR
FIN IO_ERROR

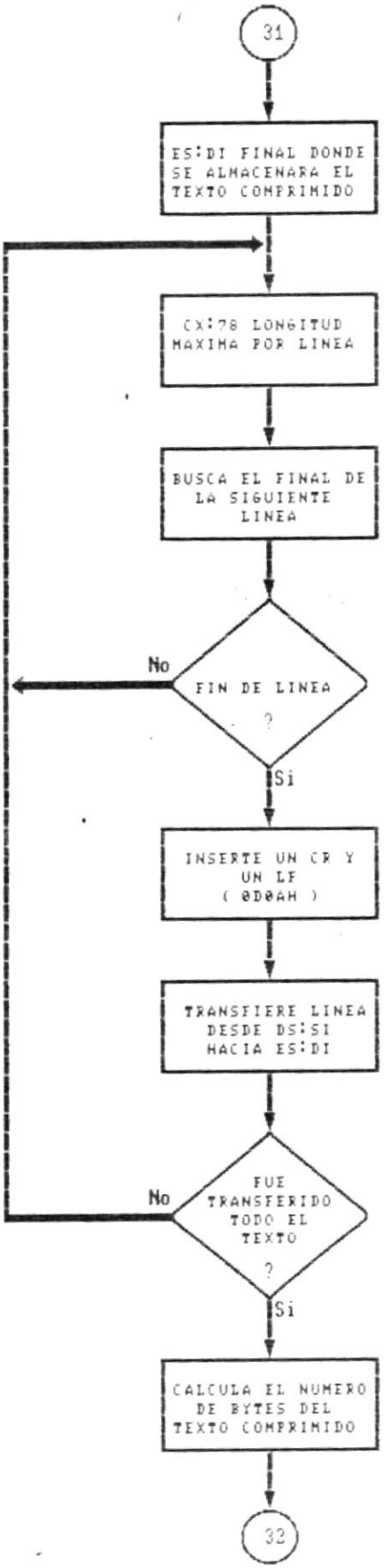

CX_MENU
FIJA PUNTERO DI EN AREA DE LAS TECLAS DEL EDITOR
AH CONTIENE EL CODIGO DE LA TECLA PRESIONADA
No
VALOR DE AH PERTENECE A ALGUNA OPCION ?
Si
EJECUTA LA SUBRUTINA DE OPCION SEGUN VALOR DE AH
RETORNA CON ZF=0 SI NO SE EJECUTO OPCION ALGUNA
RETORNA CON ZF=1 PARA INDICAR QUE SE EJECUTO OPCION
FIN CX_MENU

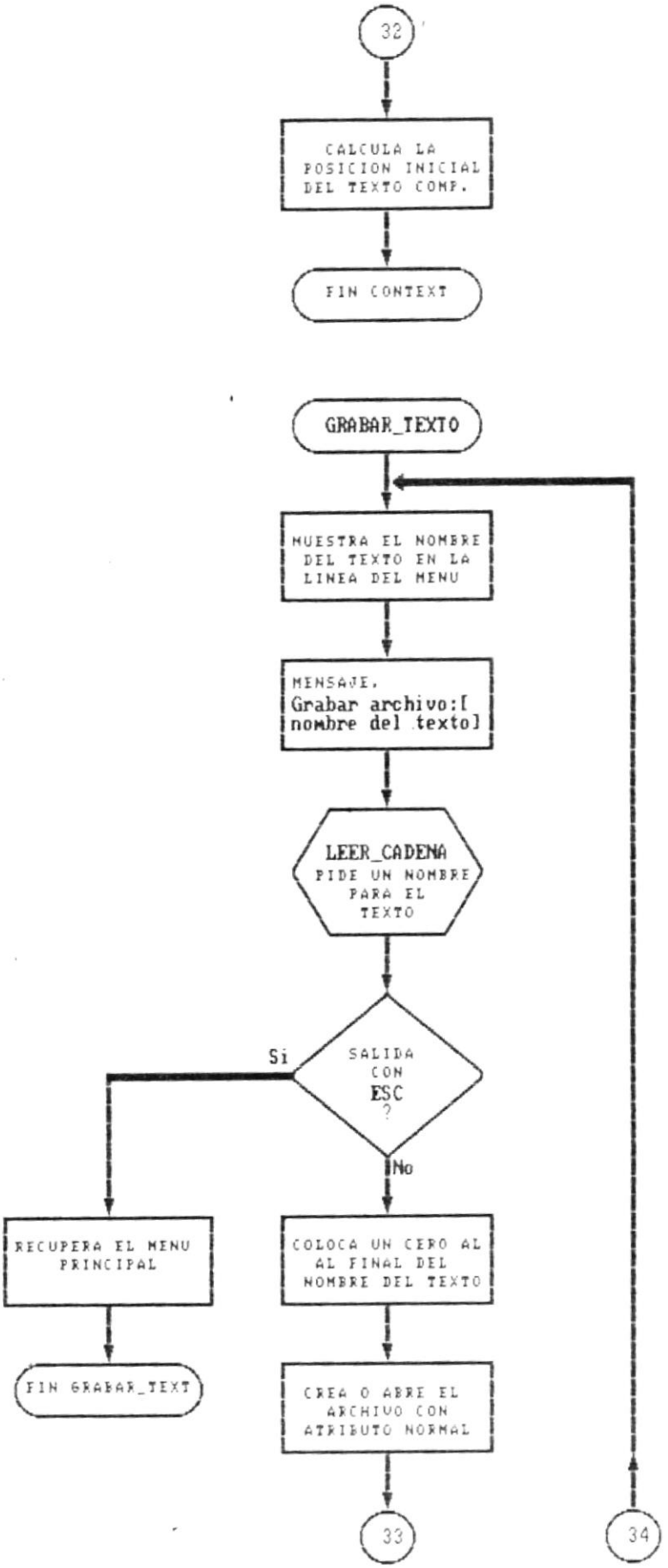

PAGINA
ES:DI EN EL AREA DE VIDEO (VENTANA EDITOR)
CX:12 * 78 CARAC.
BH: COLOR Y FONDO DE LOS CARACTERES
30

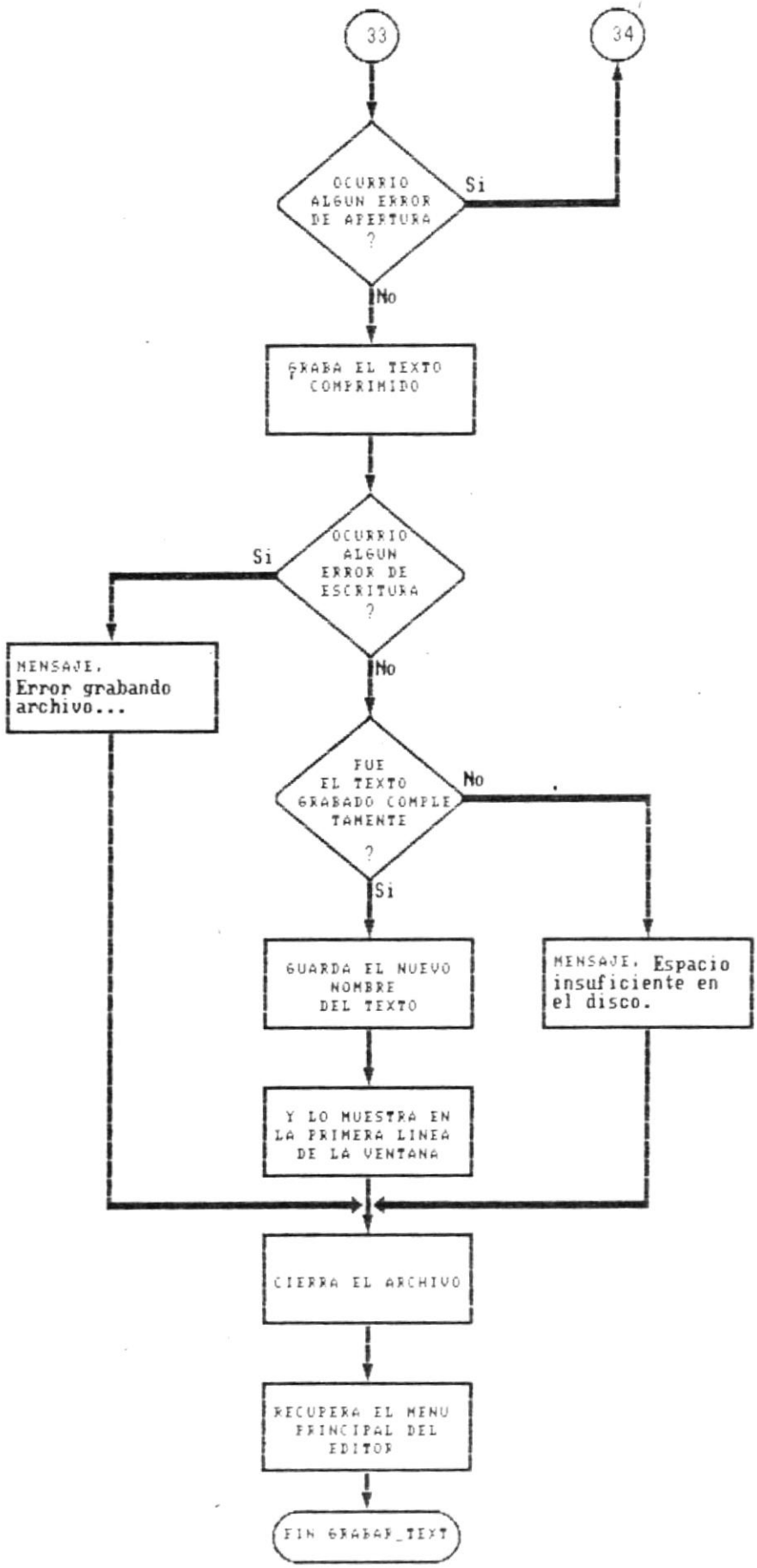

30
DS:SI EN EL AREA DE MEMORIA QUE CONTIENE EL TEXTO
TRANSFIERE LOS CX CARACTERES DESDE DS:SI HACIA ES:DI
FIN PAGINA

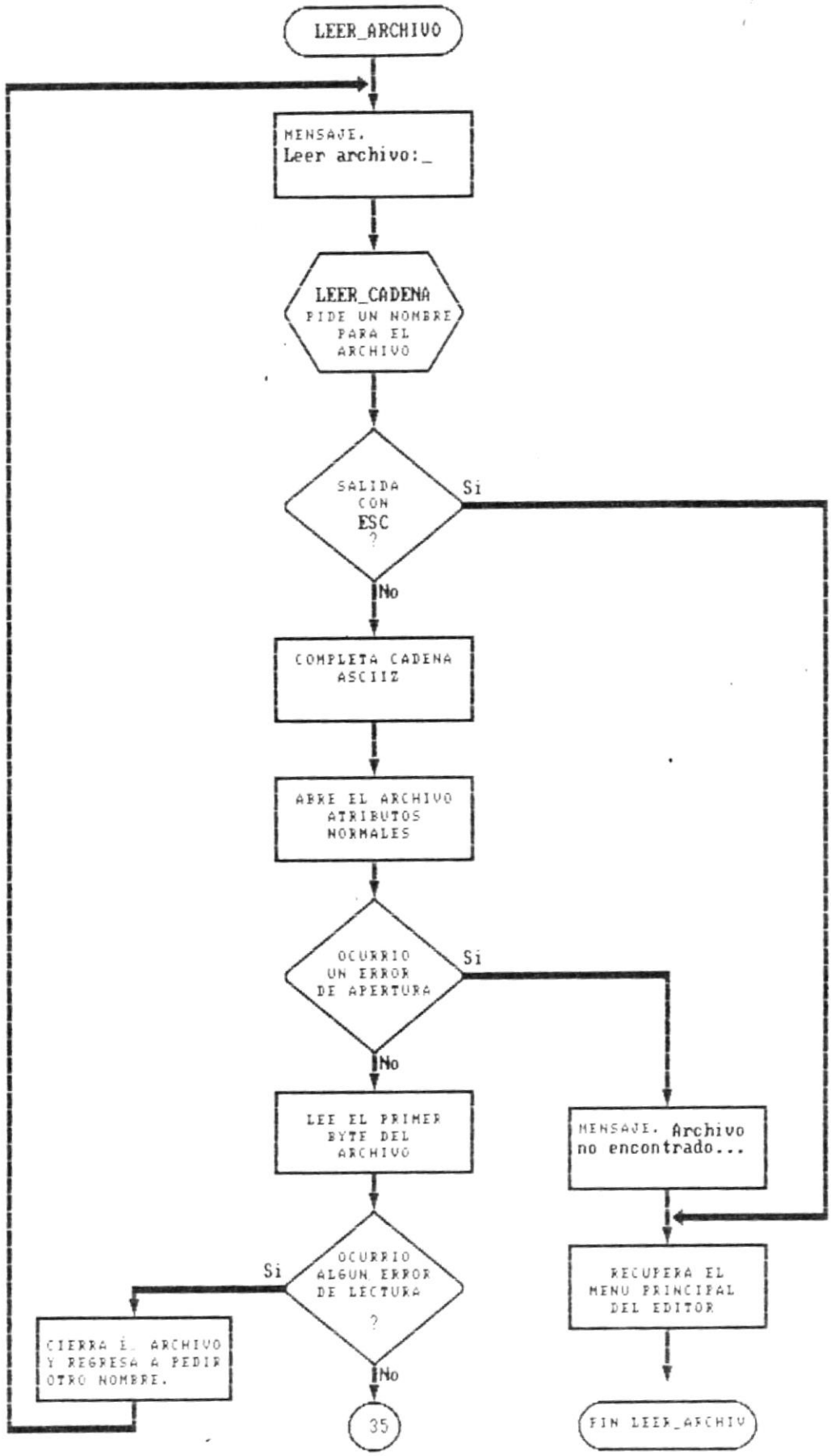

LINEA_COLUMNA
ES:DI POSICION DE VIDEO DONDE SE MUESTRA VALOR L.C
CALCULA EL VALOR DE LA COLUMNA Y LINEA DEL CURSOR
MUESTRA VALOR DE LINEA Y COLUMNA DEL CURSOR
FIN LINEA_COLUM


GRABAR
FINAL DE TEXTO
CALCULA EL FINAL DEL TEXTO
EXISTE ALGUN TEXTO PARA GRABAR ?
No
Si
CONTEXT
COMPRIME EL TEXTO
INSERTA CR - LF
GRABAR_TEXTO
TRATA DE GRABAR EL TEXTO EN EL DISCO
FIN GRABAR
CONTEXT
AJUSTA NUMERO DE LINEAS DEL TEXTO A UN VALOR ENTERO
FIJA LOS REGS. DS:SI AL FINAL DEL TEXTO
31

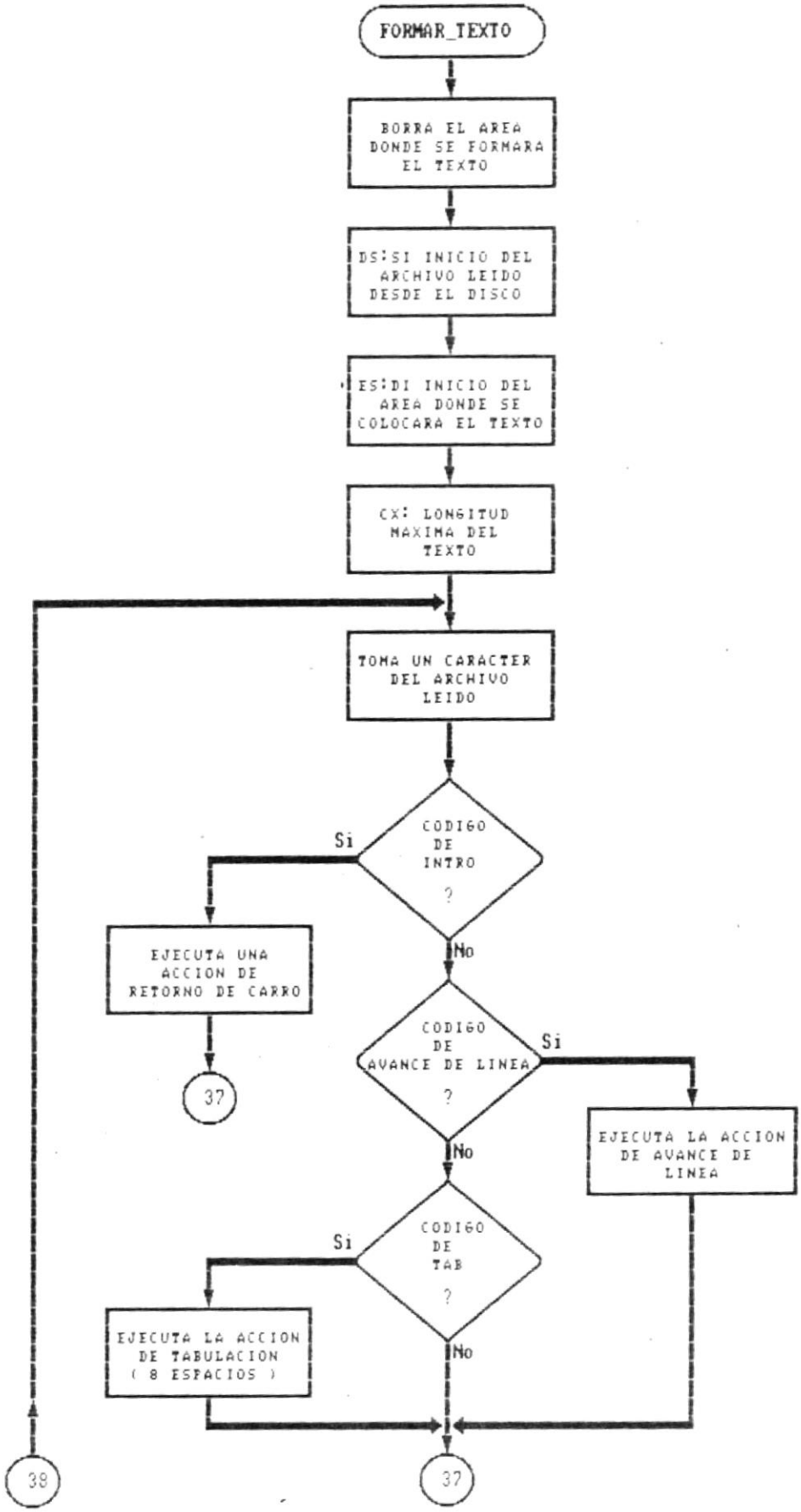

31
ES:DI FINAL DONDE SE ALMACENARA EL TEXTO COMPRIMIDO
CX:78 LONGITUD MAXIMA POR LINEA
BUSCA EL FINAL DE LA SIGUIENTE LINEA
No
FIN DE LINEA ?
Si
INSERTE UN CR Y UN LF (0D0AH)
TRANSFIERE LINEA DESDE DS:SI HACIA ES:DI
No
FUE TRANSFERIDO TODO EL TEXTO ?
Si
CALCULA EL NUMERO DE BYTES DEL TEXTO COMPRIMIDO
32

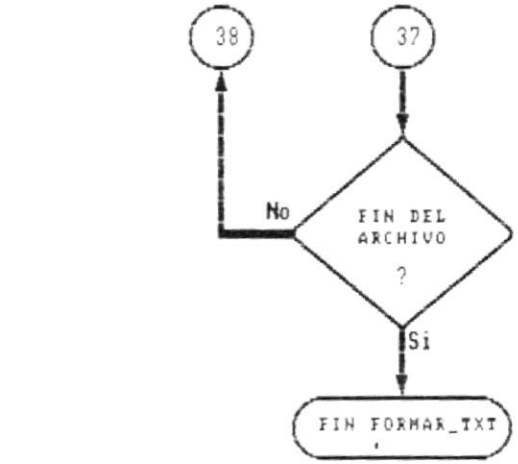

32
CALCULA LA POSICION INICIAL DEL TEXTO COMP.
FIN CONTEXT
GRABAR_TEXTO
MUESTRA EL NOMBRE DEL TEXTO EN LA LINEA DEL MENU
MENSAJE.
Grabar archivo:[nombre del texto]
LEER_CADENA
PIDE UN NOMBRE PARA EL TEXTO
SALIDA CON ESC ?
Si
No
RECUPERA EL MENU PRINCIPAL
FIN GRABAR_TEXT
COLOCA UN CERO AL AL FINAL DEL NOMBRE DEL TEXTO
CREA O ABRE EL ARCHIVO CON ATRIBUTO NORMAL
33
34

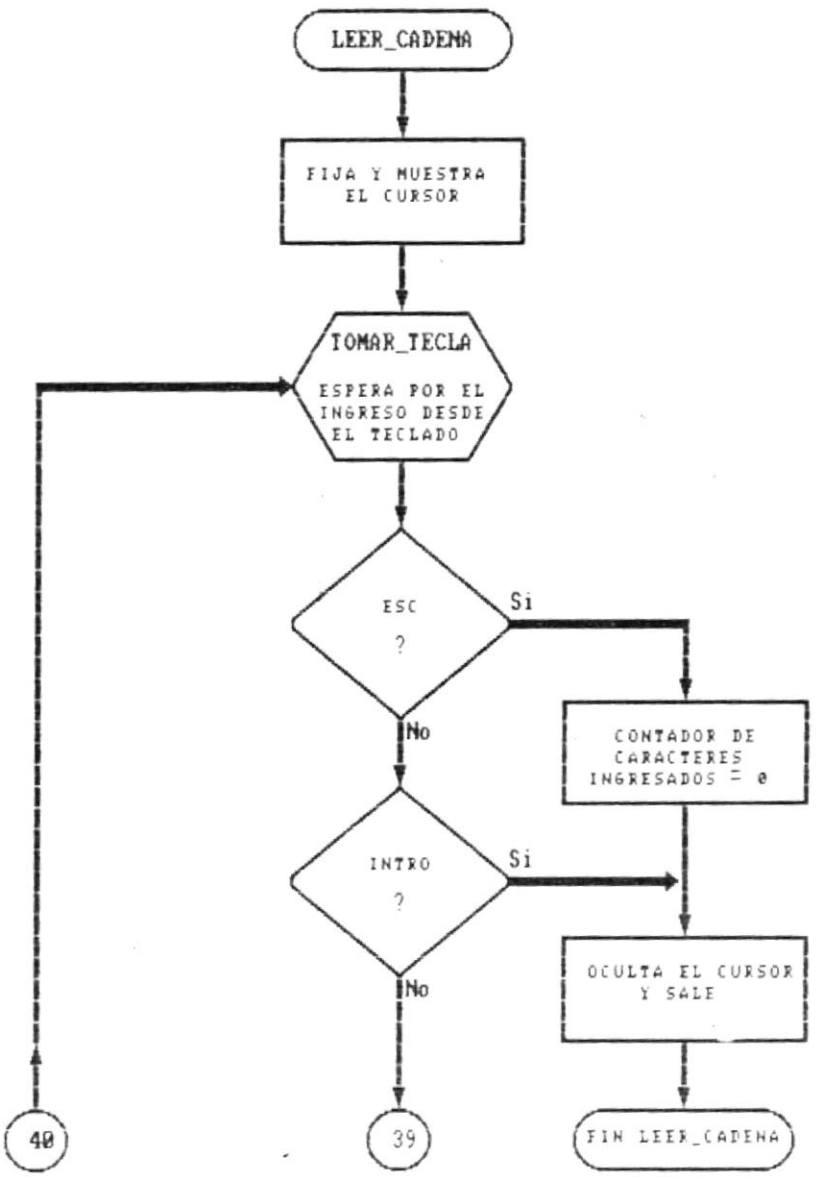

33
34
OCURRIO ALGUN ERROR DE APERTURA ?
Si
No
GRABA EL TEXTO COMPRIMIDO
OCURRIO ALGUN ERROR DE ESCRITURA ?
Si
No
MENSAJE, Error grabando archivo...
FUE EL TEXTO GRABADO COMPLETAMENTE ?
No
Si
GUARDA EL NUEVO NOMBRE DEL TEXTO
MENSAJE, Espacio insuficiente en el disco.
Y LO MUESTRA EN LA PRIMERA LINEA DE LA VENTANA
CIERRA EL ARCHIVO
RECUPERA EL MENU PRINCIPAL DEL EDITOR
FIN GRABAR_TEXT

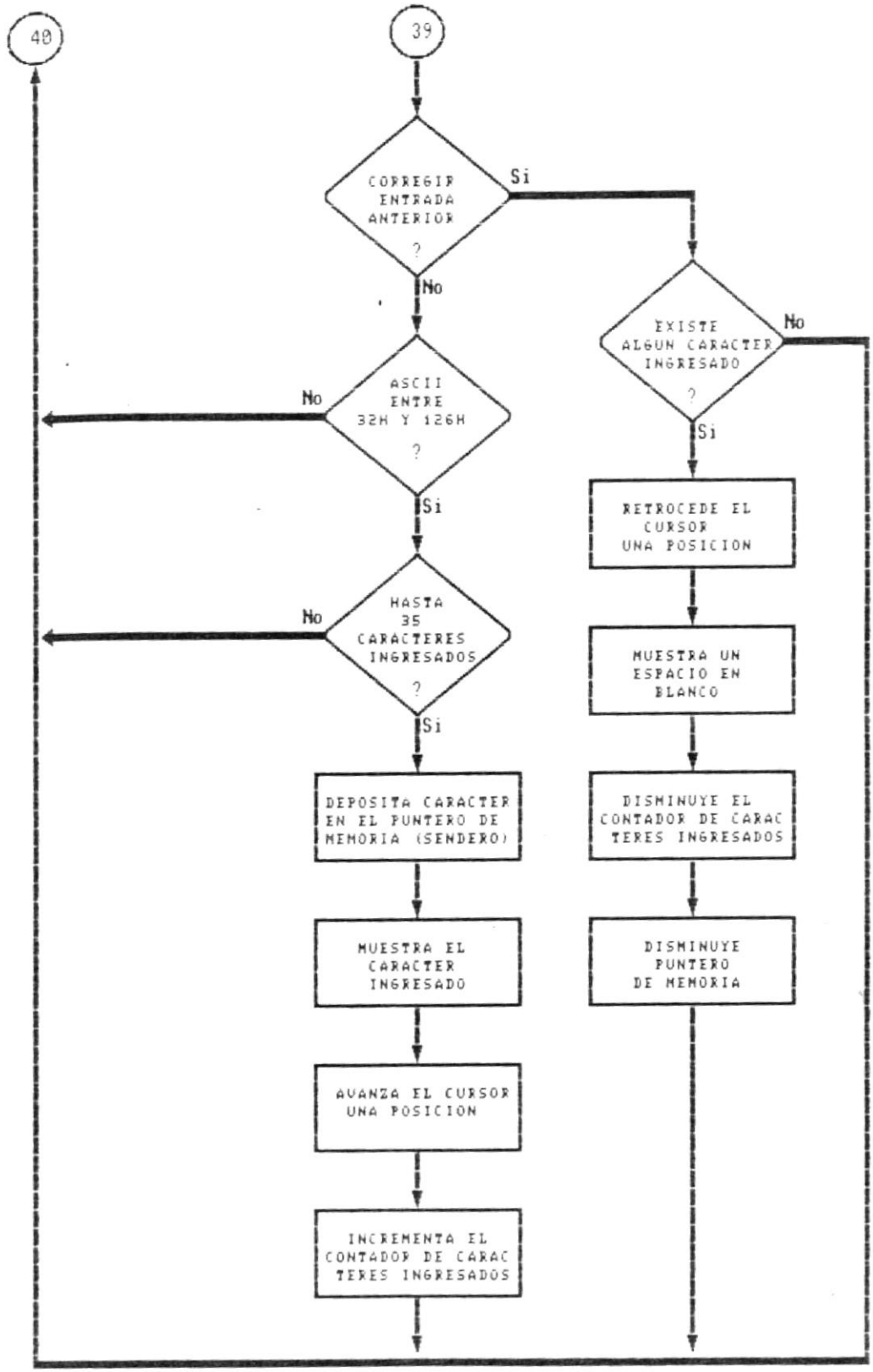

LEER_ARCHIVO
MENSAJE.
Leer archivo:_
LEER_CADENA
PIDE UN NOMBRE
PARA EL
ARCHIVO
SALIDA
CON
ESC
?
Si
No
COMPLETA CADENA
ASCIIZ
ABRE EL ARCHIVO
ATRIBUTOS
NORMALES
OCURRIO
UN ERROR
DE APERTURA
Si
No
LEE EL PRIMER
BYTE DEL
ARCHIVO
MENSAJE. Archivo
no encontrado...
OCURRIO
ALGUN ERROR
DE LECTURA
?
Si
No
CIERRA EL ARCHIVO
Y REGRESA A PEDIR
OTRO NOMBRE.
RECUPERA EL
MENU PRINCIPAL
DEL EDITOR
35
FIN LEER_ARCHIV

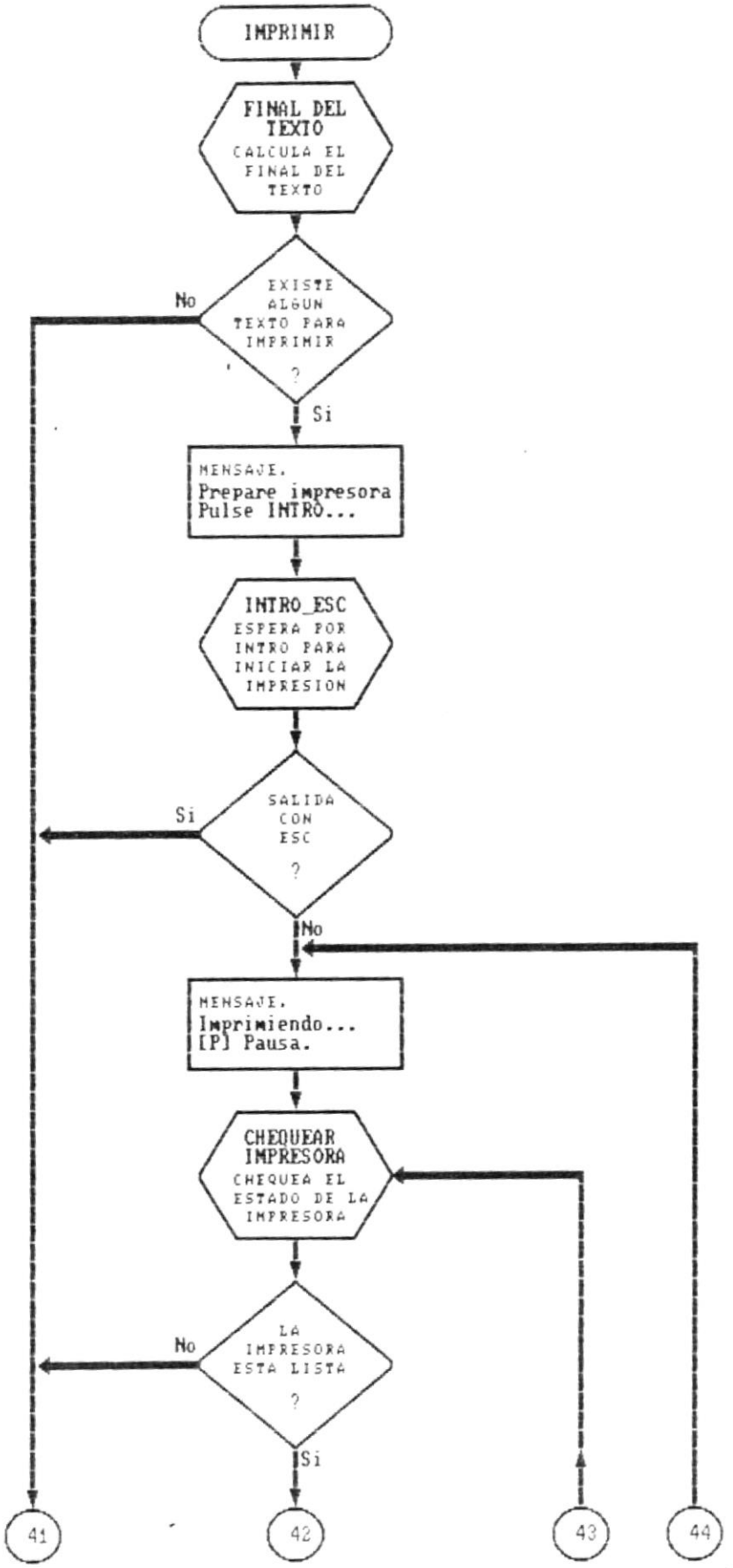

35
LEE HASTA 7800 CARACTERES DEL ARCHIV O
CIERRA EL ARCHIVO
FORMAR_TEXTO
ELIMINA LOS CARACTERES CR - LF
GUARDA EL NOMBRE DEL ARCHIVO LEIDO
Y LO MUESTRA EN LA PRIMERA LINEA DE LA VENTANA
ACTUALIZA EL MENU PRINCIPAL DEL EDITOR
FIN LEER_ARCH.

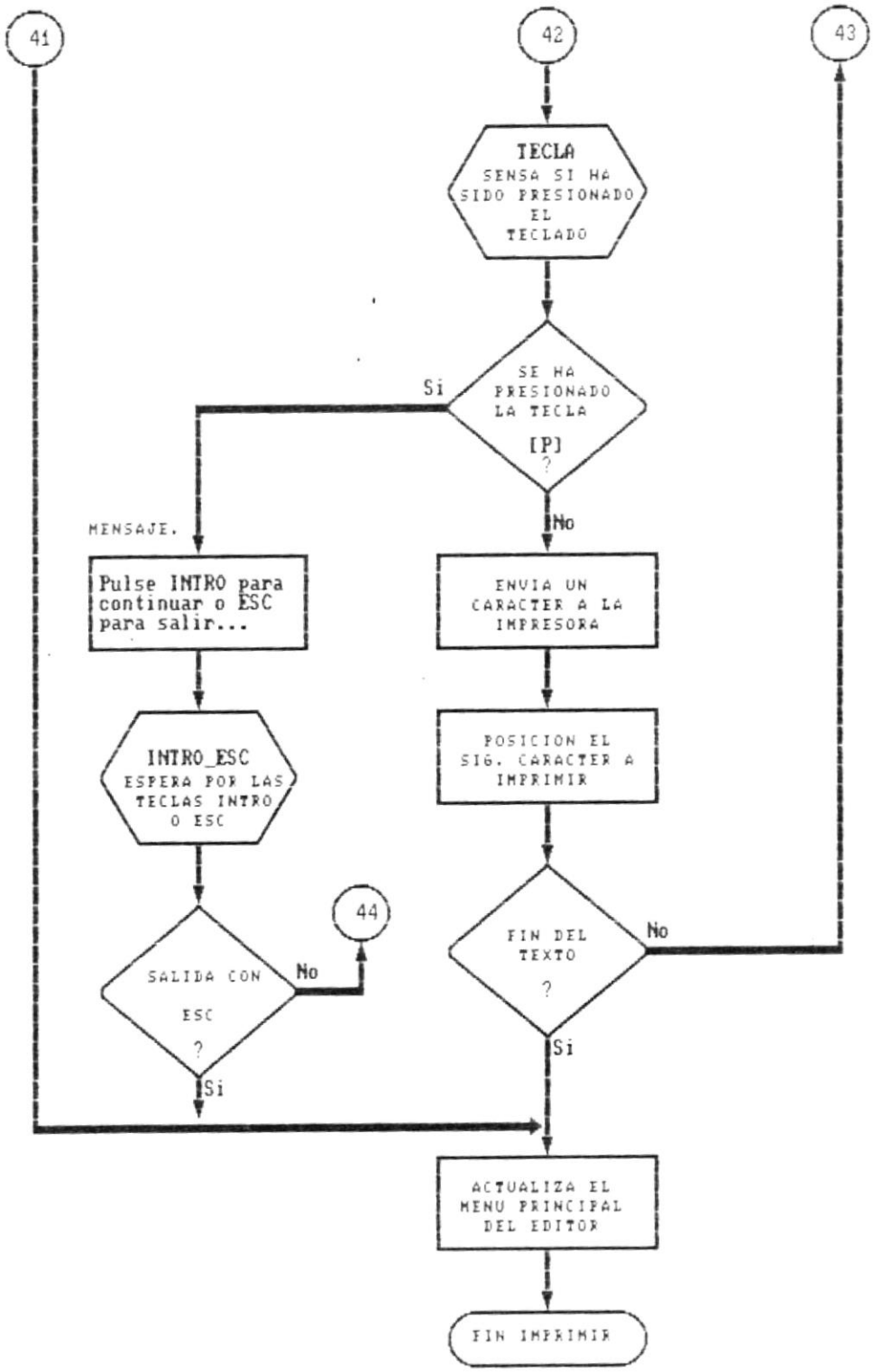

FORMAR_TEXTO
BORRA EL AREA DONDE SE FORMARA EL TEXTO
DS:SI INICIO DEL ARCHIVO LEIDO DESDE EL DISCO
ES:DI INICIO DEL AREA DONDE SE COLOCARA EL TEXTO
CX: LONGITUD MAXIMA DEL TEXTO
TOMA UN CARACTER DEL ARCHIVO LEIDO
CODIGO DE INTRO ?
Si
No
EJECUTA UNA ACCION DE RETORNO DE CARRO
37
CODIGO DE AVANCE DE LINEA ?
Si
No
EJECUTA LA ACCION DE AVANCE DE LINEA
CODIGO DE TAB ?
Si
No
EJECUTA LA ACCION DE TABULACION (8 ESPACIOS)
38
37

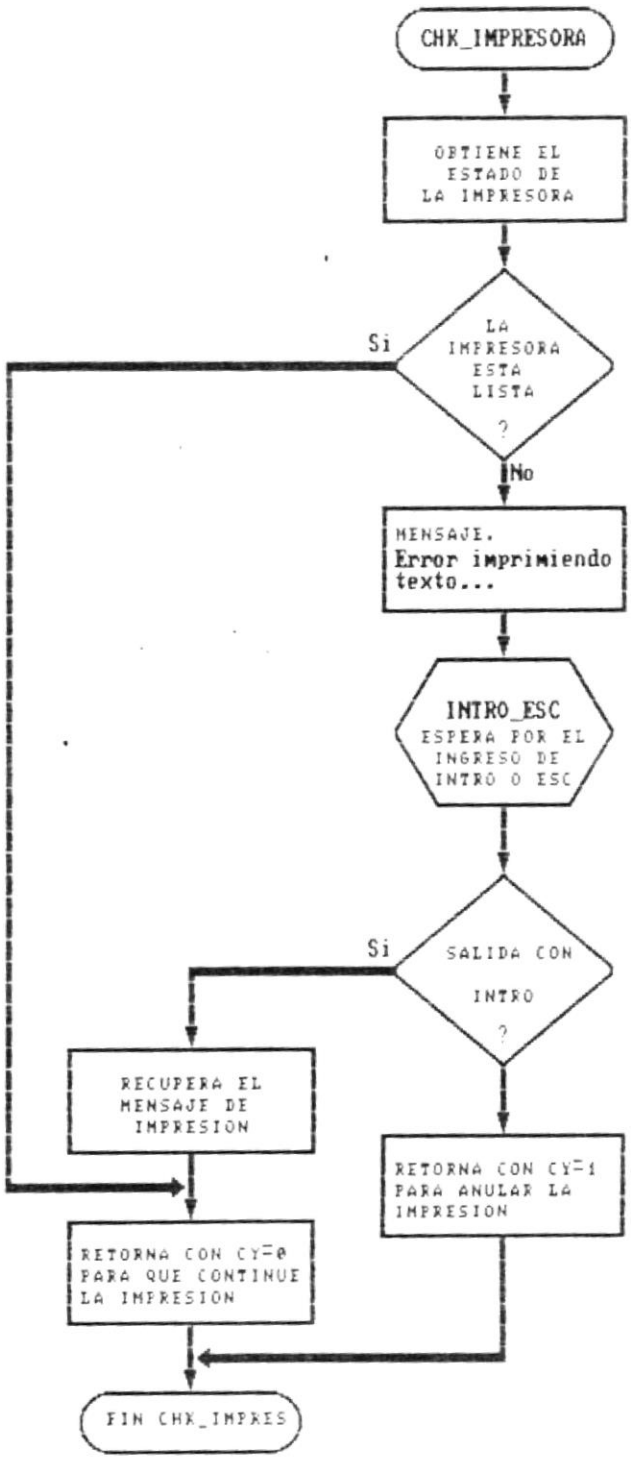

38
37
No
FIN DEL ARCHIVO ?
Si
FIN FORMAR_TXT

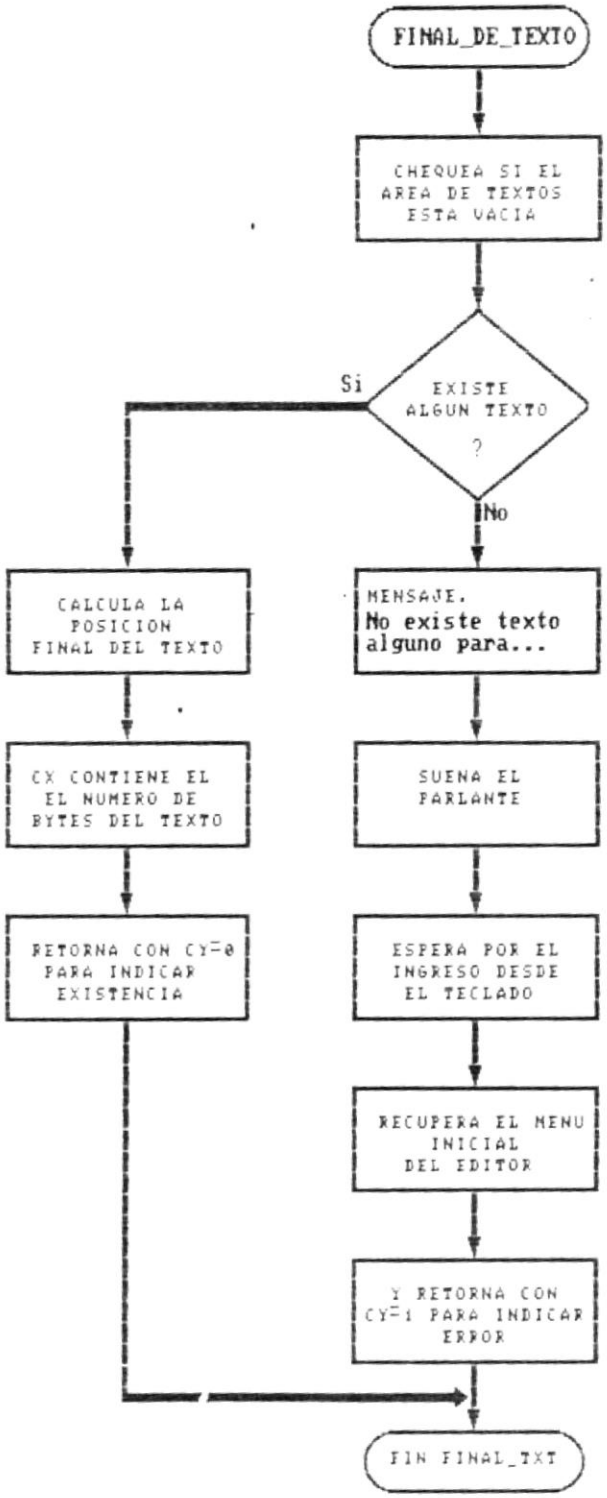

LEER_CADENA
FIJA Y MUESTRA EL CURSOR
TOMAR_TECLA
ESPERA POR EL INGRESO DESDE EL TECLADO
ESC ?
Si
No
CONTADOR DE CARACTERES INGRESADOS = 0
INTRO ?
Si
No
OCULTA EL CURSOR Y SALE
40
39
FIN LEER_CADENA

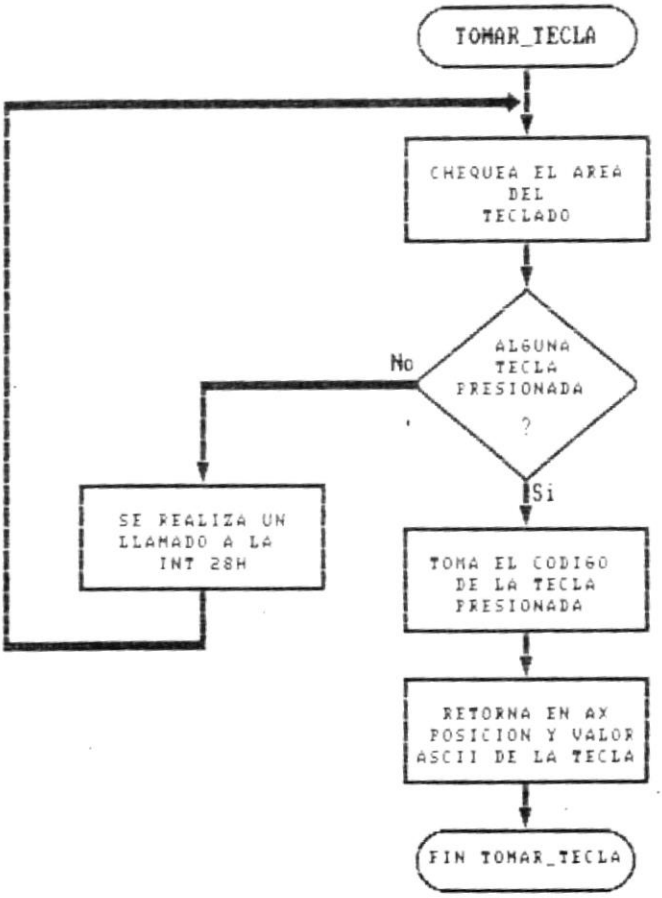

40
39
CORREGIR ENTRADA ANTERIOR ?
Si
No
EXISTE ALGUN CARACTER INGRESADO ?
No
Si
ASCII ENTRE 32H Y 126H ?
No
Si
RETROCEDE EL CURSOR UNA POSICION
HASTA 35 CARACTERES INGRESADOS ?
No
Si
MUESTRA UN ESPACIO EN BLANCO
DEPOSITA CARACTER EN EL PUNTERO DE MEMORIA (SENDERO)
DISMINUYE EL CONTADOR DE CARACTERES INGRESADOS
MUESTRA EL CARACTER INGRESADO
DISMINUYE PUNTERO DE MEMORIA
AVANZA EL CURSOR UNA POSICION
INCREMENTA EL CONTADOR DE CARACTERES INGRESADOS

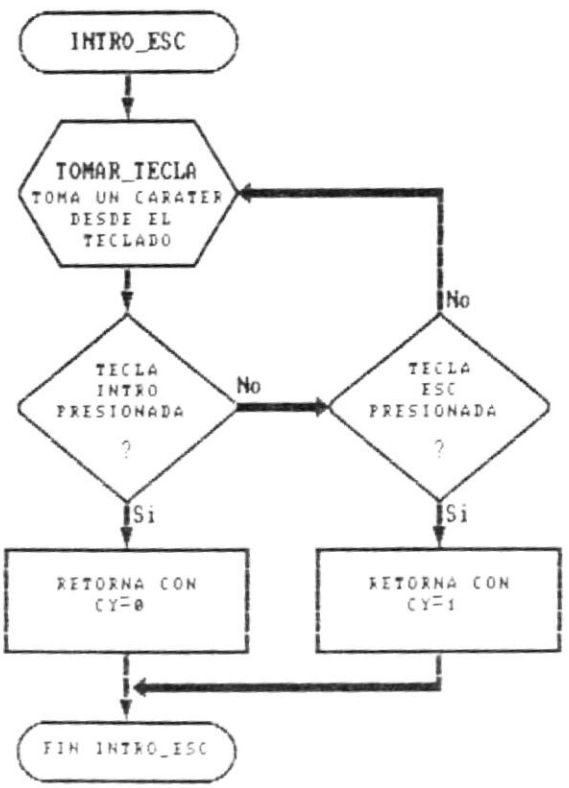

IMPRIMIR
FINAL DEL TEXTO
CALCULA EL FINAL DEL TEXTO
EXISTE ALGUN TEXTO PARA IMPRIMIR ?
No
Si
MENSAJE.
Prepare impresora
Pulse INTRO...
INTRO_ESC
ESPERA POR INTRO PARA INICIAR LA IMPRESION
SALIDA CON ESC ?
Si
No
MENSAJE.
Imprimiendo...
[P] Pausa.
CHEQUEAR IMPRESORA
CHEQUEA EL ESTADO DE LA IMPRESORA
LA IMPRESORA ESTA LISTA ?
No
Si
41
42
43
44

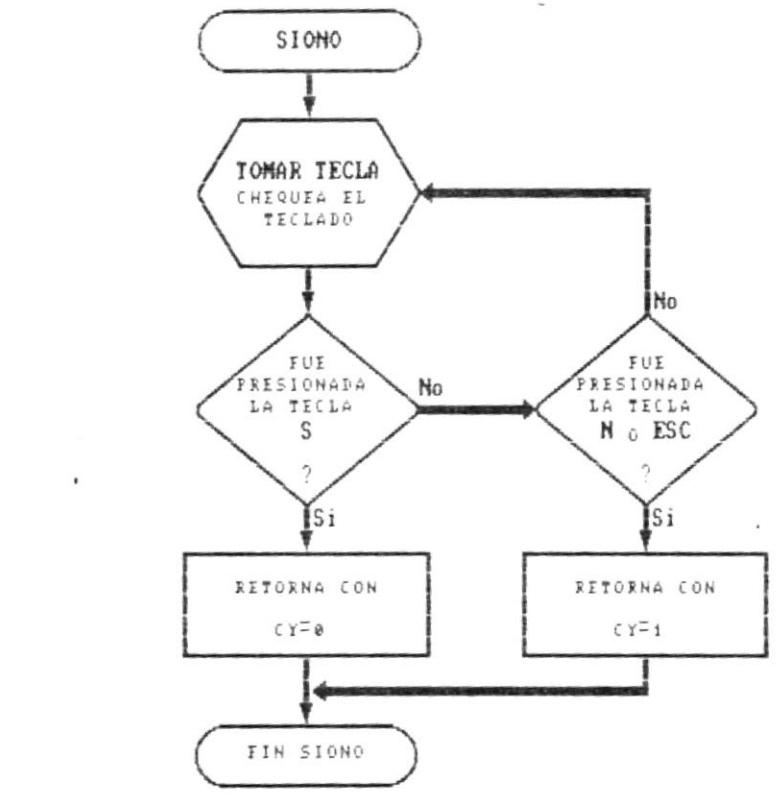

41
42
43
TECLA
SENSA SI HA SIDO PRESIONADO EL TECLADO
SE HA PRESIONADO LA TECLA [P] ?
Si
No
MENSAJE.
Pulse INTRO para continuar o ESC para salir...
ENVIA UN CARACTER A LA IMPRESORA
INTRO_ESC
ESPERA POR LAS TECLAS INTRO O ESC
POSICION EL SIG. CARACTER A IMPRIMIR
44
SALIDA CON ESC ?
No
Si
FIN DEL TEXTO ?
No
Si
ACTUALIZA EL MENU PRINCIPAL DEL EDITOR
FIN IMPRIMIR

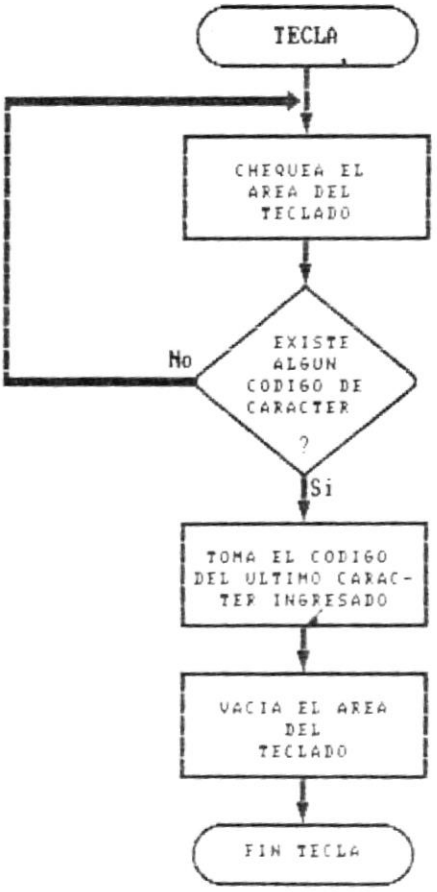

CHK_IMPRESORA
OBTIENE EL ESTADO DE LA IMPRESORA
LA IMPRESORA ESTA LISTA ?
Si
No
MENSAJE. Error imprimiendo texto...
INTRO_ESC ESPERA POR EL INGRESO DE INTRO O ESC
SALIDA CON INTRO ?
Si
RECUPERA EL MENSAJE DE IMPRESION
RETORNA CON CY=1 PARA ANULAR LA IMPRESION
RETORNA CON CY=0 PARA QUE CONTINUE LA IMPRESION
FIN CHK_IMPRES

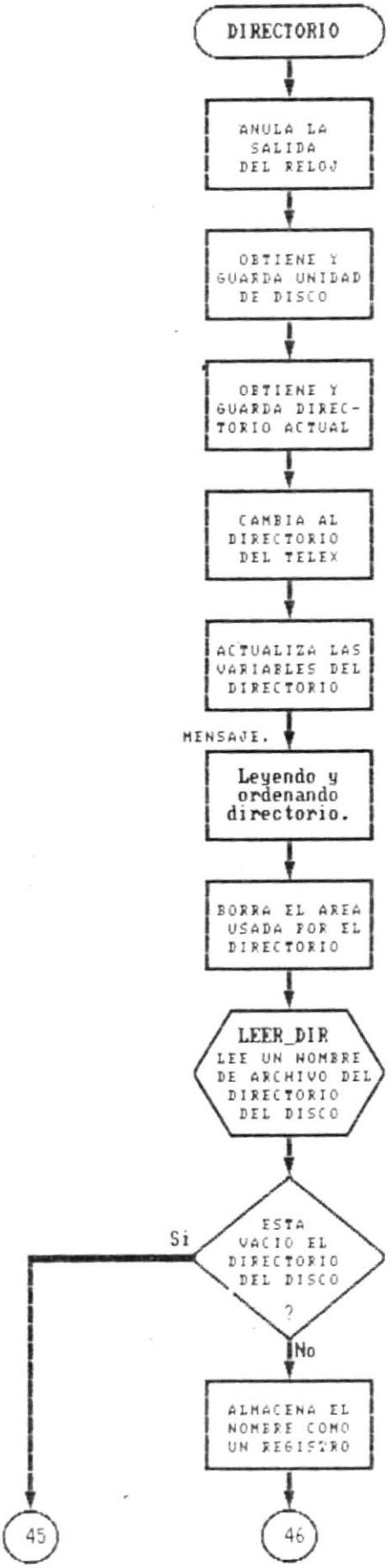

FINAL_DE_TEXTO
CHEQUEA SI EL AREA DE TEXTOS ESTA VACIA
EXISTE ALGUN TEXTO ?
Si
No
CALCULA LA POSICION FINAL DEL TEXTO
CX CONTIENE EL EL NUMERO DE BYTES DEL TEXTO
RETORNA CON CY=0 PARA INDICAR EXISTENCIA
MENSAJE. No existe texto alguno para...
SUENA EL PARLANTE
ESPERA POR EL INGRESO DESDE EL TECLADO
RECUPERA EL MENU INICIAL DEL EDITOR
Y RETORNA CON CY=1 PARA INDICAR ERROR
FIN FINAL_TXT

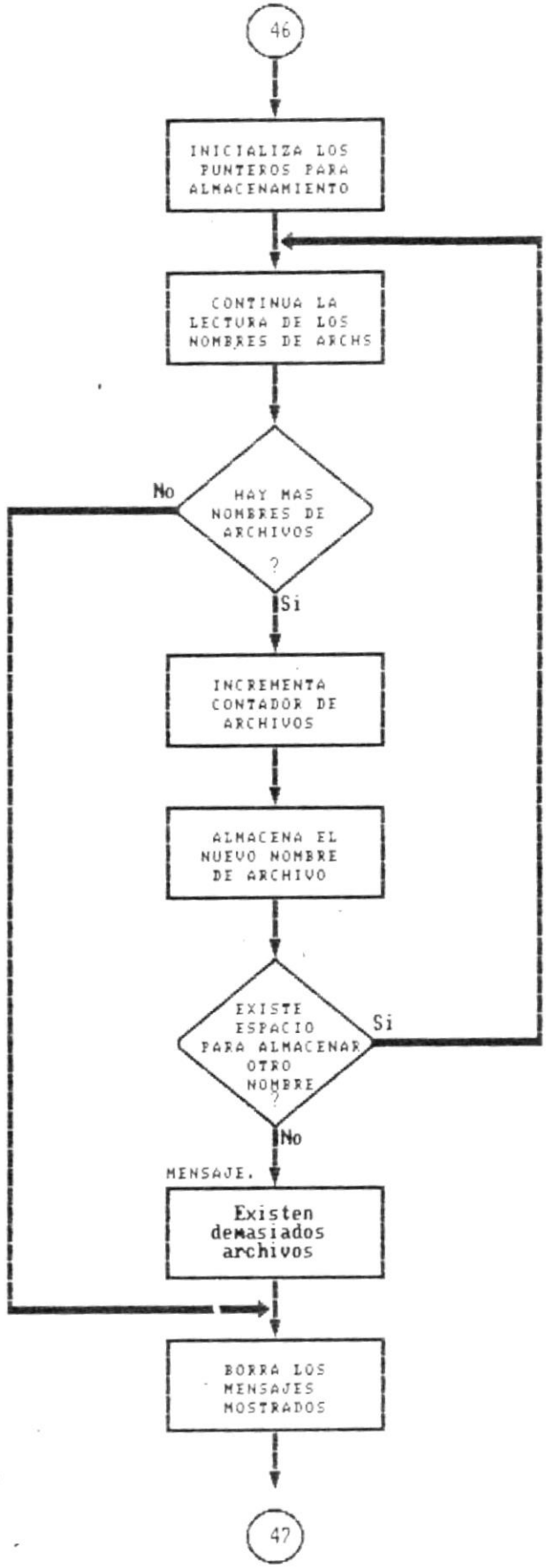

TOMAR_TECLA
CHEQUEA EL AREA DEL TECLADO
ALGUNA TECLA PRESIONADA ?
No
Si
SE REALIZA UN LLAMADO A LA INT 28H
TOMA EL CODIGO DE LA TECLA PRESIONADA
RETORNA EN AX POSICION Y VALOR ASCII DE LA TECLA
FIN TOMAR_TECLA

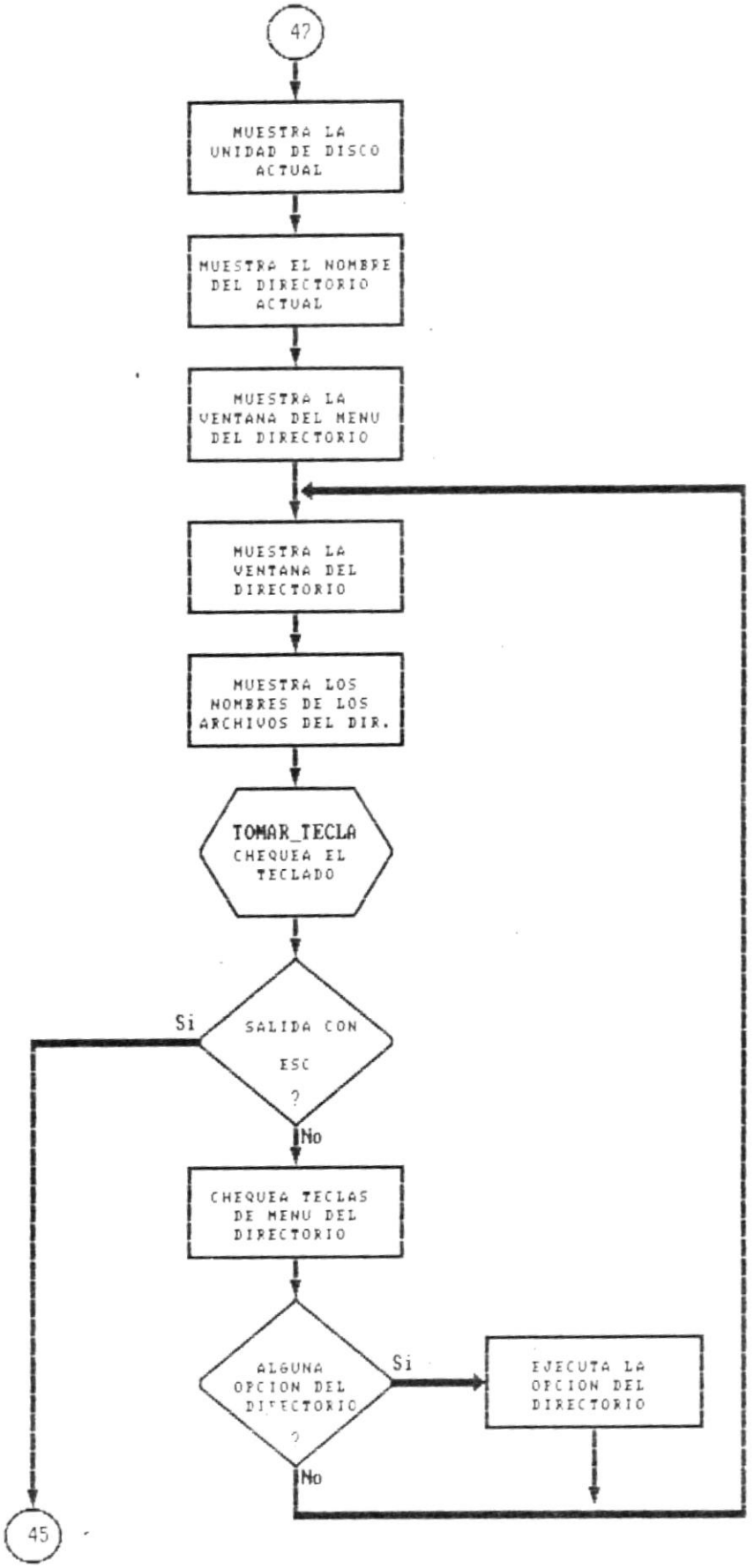

INTRO_ESC
TOMAR_TECLA
TOMA UN CARATER DESDE EL TECLADO
TECLA INTRO PRESIONADA ?
No
TECLA ESC PRESIONADA ?
No
Si
Si
RETORNA CON CY=0
RETORNA CON CY=1
FIN INTRO_ESC

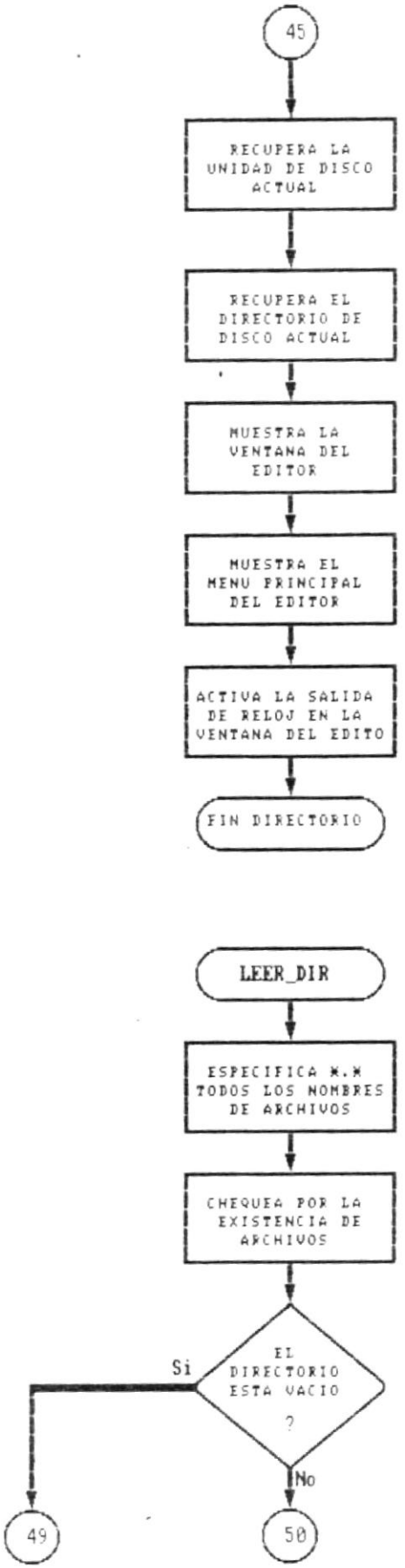

SIONO
TOMAR TECLA
CHEQUEA EL TECLADO
FUE PRESIONADA LA TECLA S ?
No
FUE PRESIONADA LA TECLA N o ESC ?
No
Si
Si
RETORNA CON CY=0
RETORNA CON CY=1
FIN SIONO

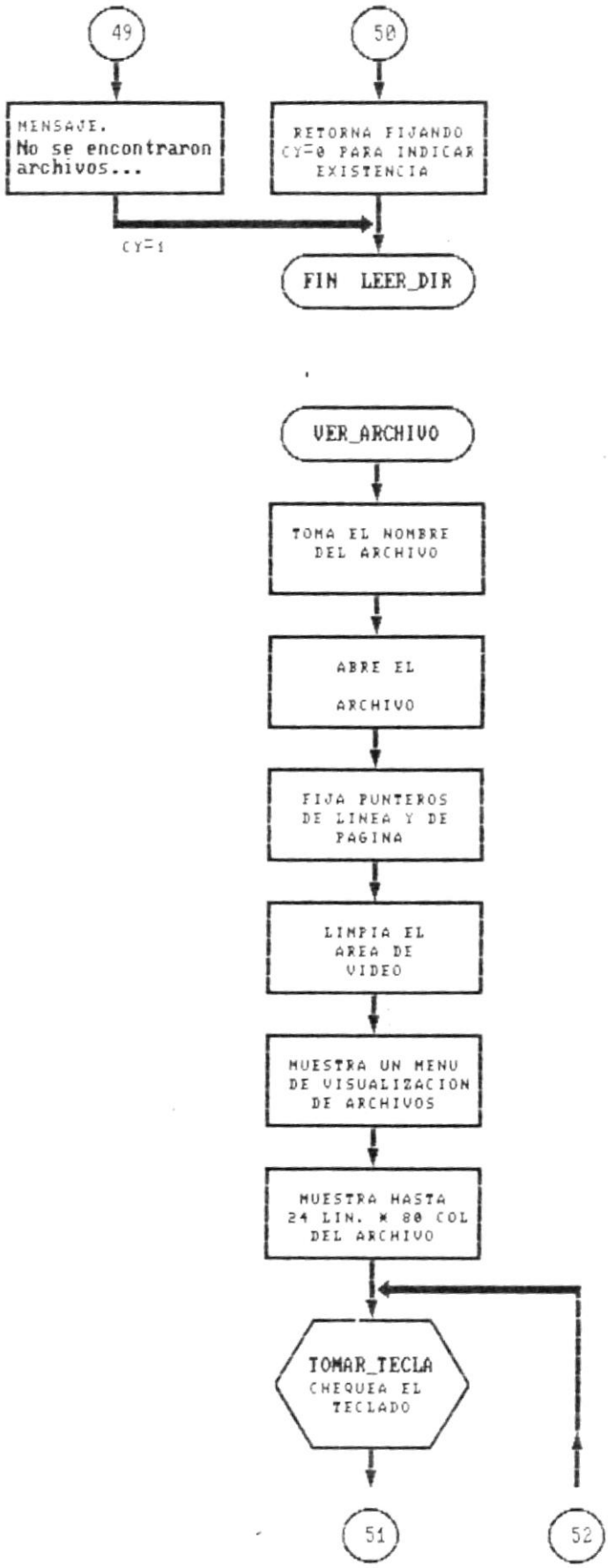

TECLA
CHEQUEA EL AREA DEL TECLADO
EXISTE ALGUN CODIGO DE CARACTER ?
No
Si
TOMA EL CODIGO DEL ULTIMO CARACTER INGRESADO
VACIA EL AREA DEL TECLADO
FIN TECLA

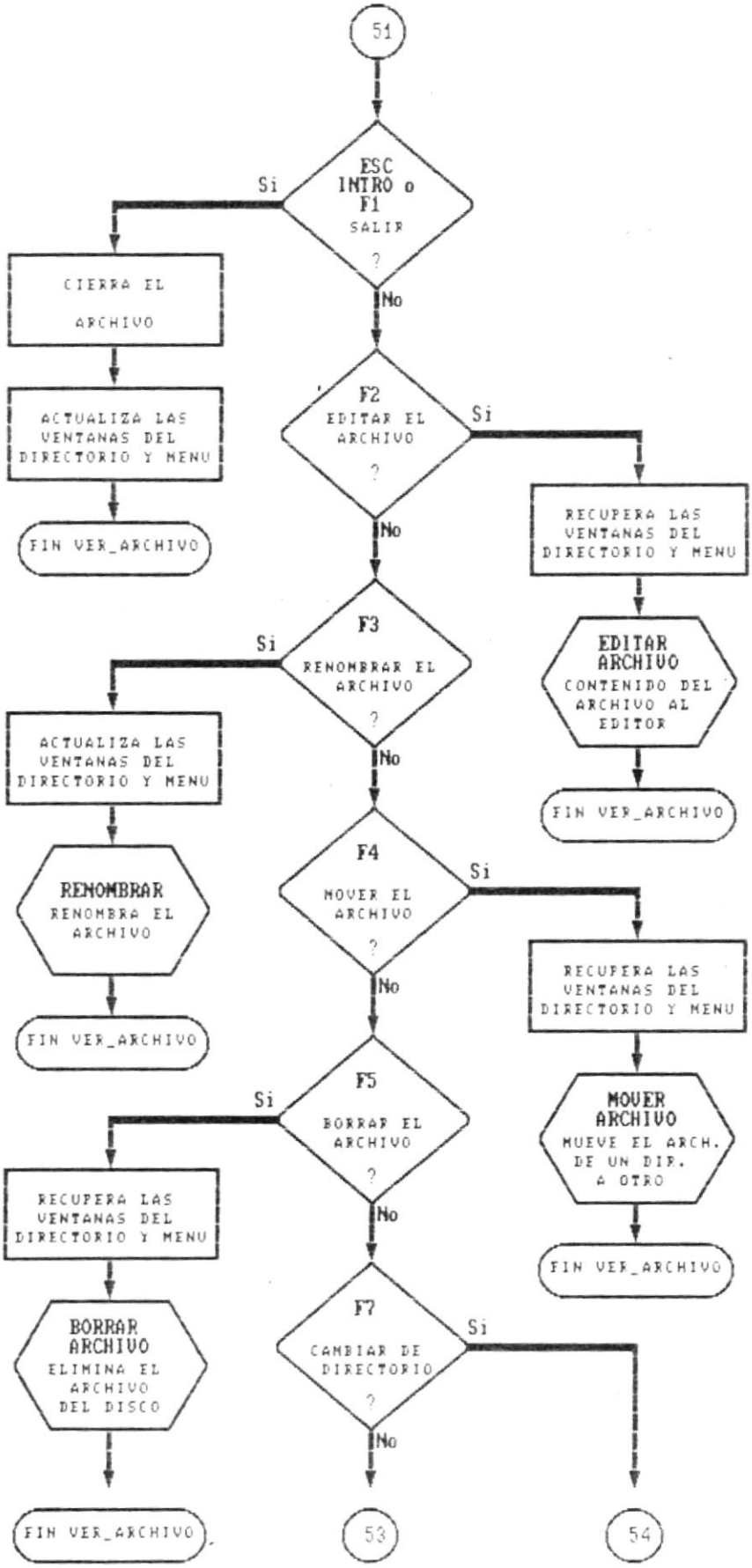

DIRECTORIO
ANULA LA SALIDA DEL RELOJ
OBTIENE Y GUARDA UNIDAD DE DISCO
OBTIENE Y GUARDA DIRECTORIO ACTUAL
CAMBIA AL DIRECTORIO DEL TELEX
ACTUALIZA LAS VARIABLES DEL DIRECTORIO
MENSAJE.
Leyendo y ordenando directorio.
BORRA EL AREA USADA POR EL DIRECTORIO
LEER_DIR
LEE UN NOMBRE DE ARCHIVO DEL DIRECTORIO DEL DISCO
ESTA VACIO EL DIRECTORIO DEL DISCO ?
Si
No
ALMACENA EL NOMBRE COMO UN REGISTRO
45
46

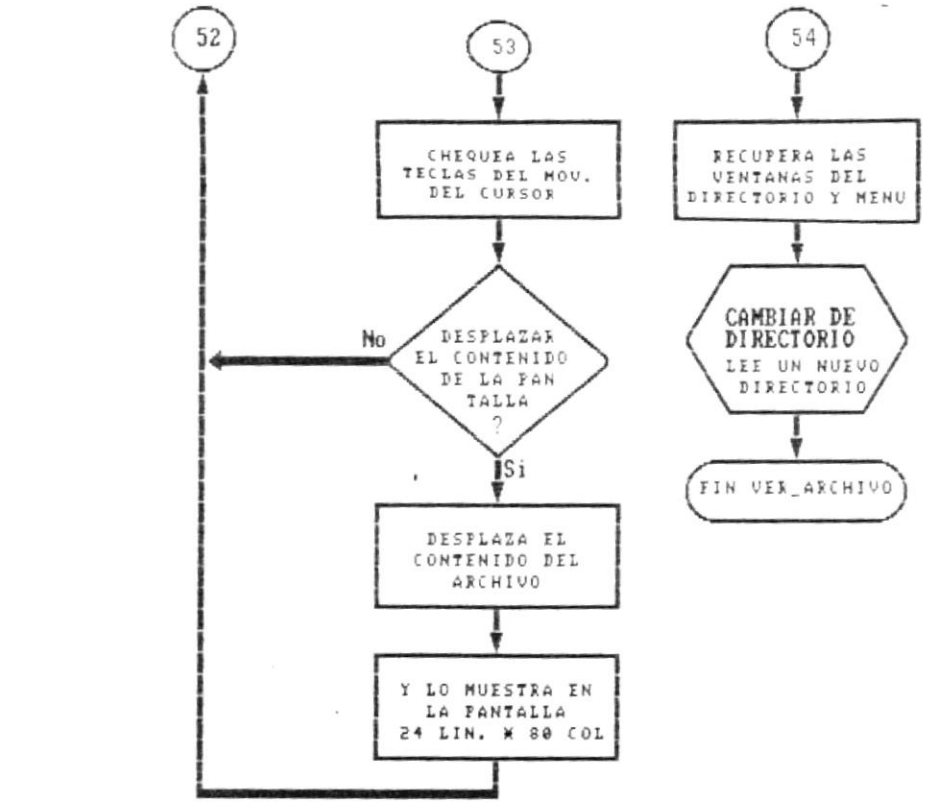

46
INICIALIZA LOS PUNTEROS PARA ALMACENAMIENTO
CONTINUA LA LECTURA DE LOS NOMBRES DE ARCHS
No
HAY MAS NOMBRES DE ARCHIVOS ?
Si
INCREMENTA CONTADOR DE ARCHIVOS
ALMACENA EL NUEVO NOMBRE DE ARCHIVO
EXISTE ESPACIO PARA ALMACENAR OTRO NOMBRE ?
Si
No
MENSAJE.
Existen demasiados archivos
BORRA LOS MENSAJES MOSTRADOS
47

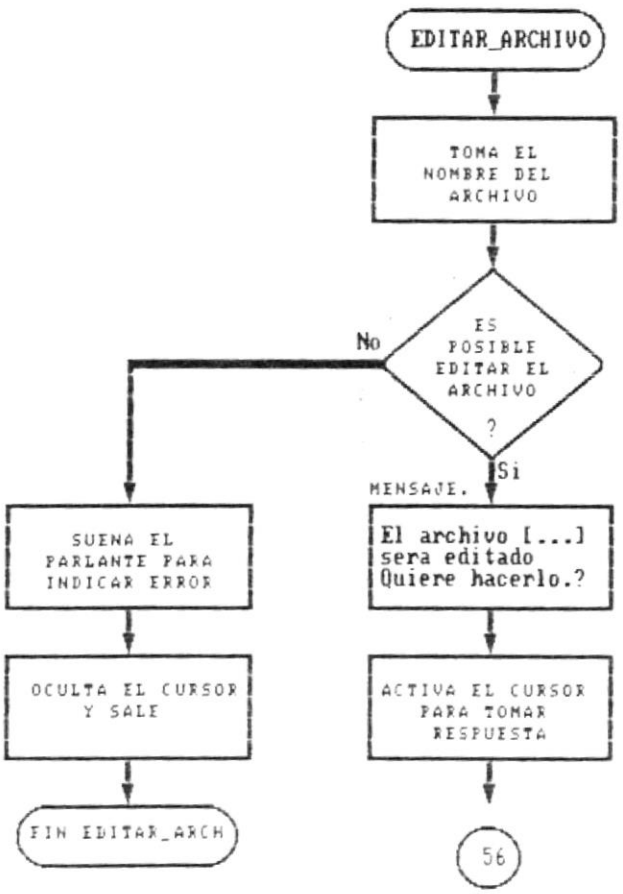

47
MUESTRA LA UNIDAD DE DISCO ACTUAL
MUESTRA EL NOMBRE DEL DIRECTORIO ACTUAL
MUESTRA LA VENTANA DEL MENU DEL DIRECTORIO
MUESTRA LA VENTANA DEL DIRECTORIO
MUESTRA LOS NOMBRES DE LOS ARCHIVOS DEL DIR.
TOMAR_TECLA CHEQUEA EL TECLADO
SALIDA CON ESC ?
Si
No
CHEQUEA TECLAS DE MENU DEL DIRECTORIO
ALGUNA OPCION DEL DIRECTORIO ?
Si
No
EJECUTA LA OPCION DEL DIRECTORIO
45

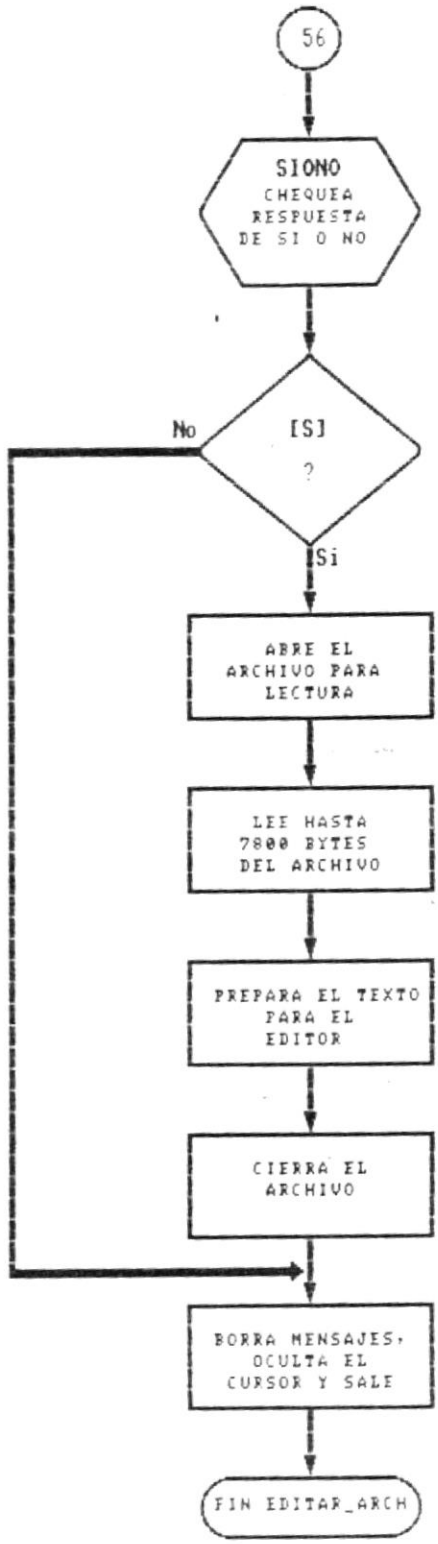

45
RECUPERA LA UNIDAD DE DISCO ACTUAL
RECUPERA EL DIRECTORIO DE DISCO ACTUAL
MUESTRA LA VENTANA DEL EDITOR
MUESTRA EL MENU PRINCIPAL DEL EDITOR
ACTIVA LA SALIDA DE RELOJ EN LA VENTANA DEL EDITO
FIN DIRECTORIO
LEER_DIR
ESPECIFICA *.* TODOS LOS NOMBRES DE ARCHIVOS
CHEQUEA POR LA EXISTENCIA DE ARCHIVOS
EL DIRECTORIO ESTA VACIO ?
Si
No
49
50

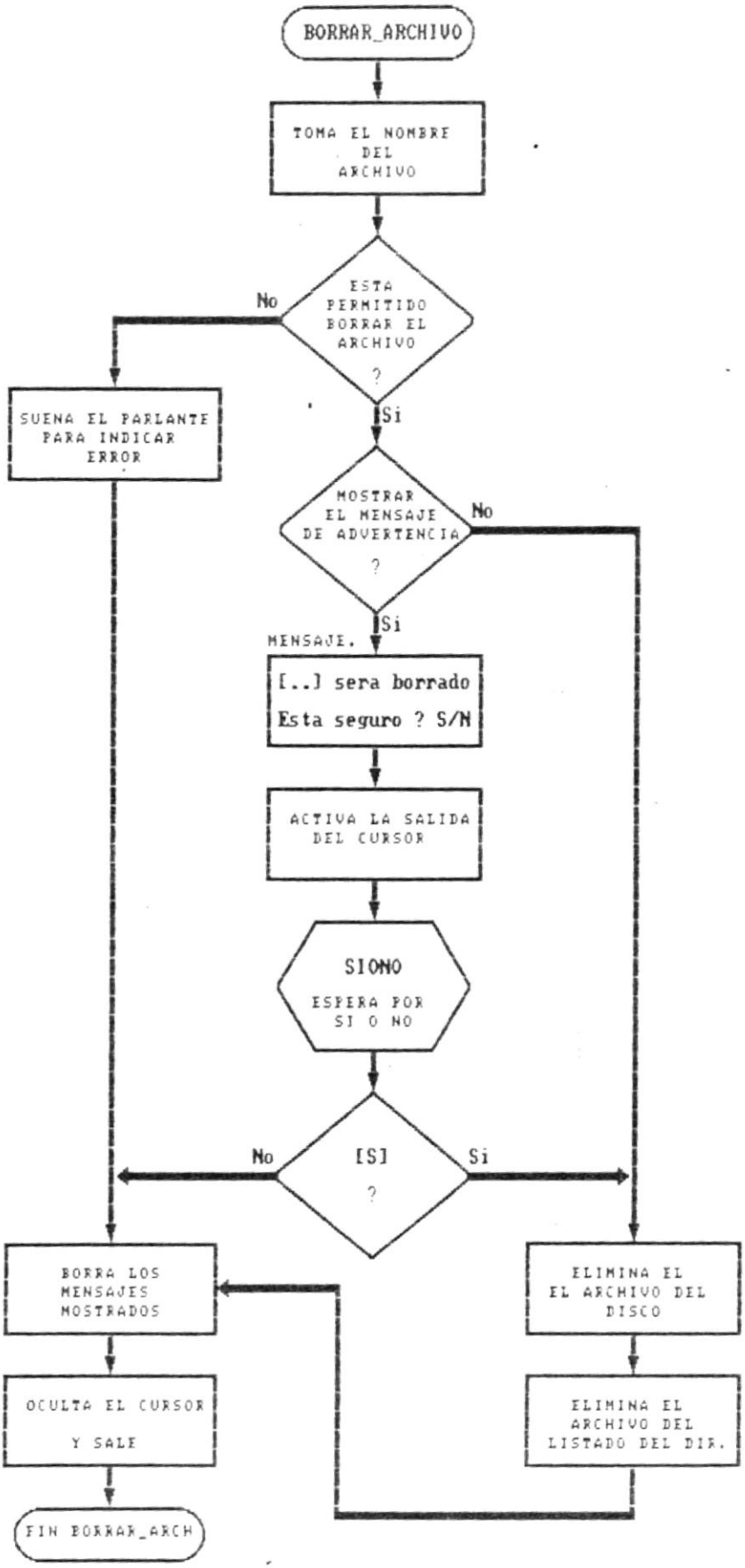

49
50
MENSAJE.
No se encontraron archivos...
RETORNA FIJANDO CY=0 PARA INDICAR EXISTENCIA
CY=1
FIN LEER_DIR
VER_ARCHIVO
TOMA EL NOMBRE DEL ARCHIVO
ABRE EL ARCHIVO
FIJA PUNTEROS DE LINEA Y DE PAGINA
LIMPIA EL AREA DE VIDEO
MUESTRA UN MENU DE VISUALIZACION DE ARCHIVOS
MUESTRA HASTA 24 LIN. X 80 COL DEL ARCHIVO
TOMAR_TECLA
CHEQUEA EL TECLADO
51
52

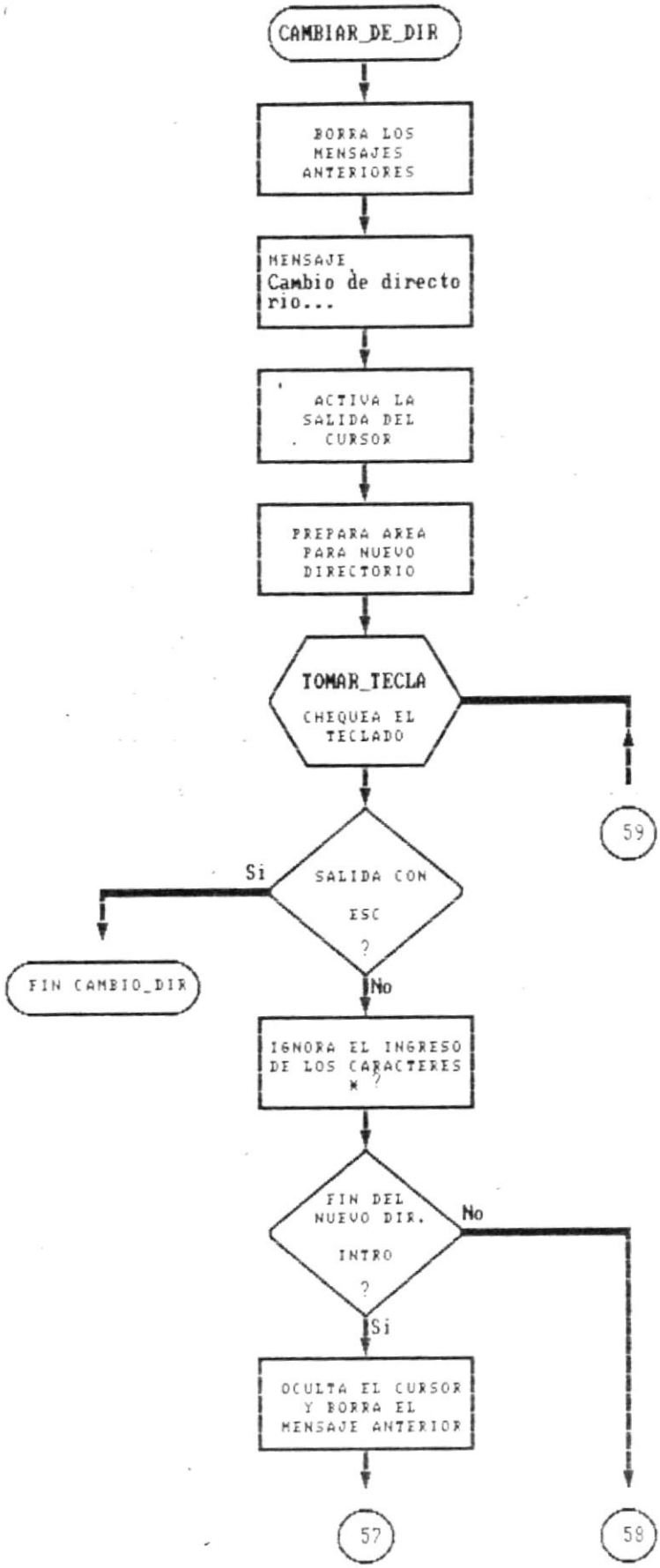

51
ESC
INTRO o
F1
SALIR
?
Si
No
CIERRA EL
ARCHIVO
ACTUALIZA LAS
VENTANAS DEL
DIRECTORIO Y MENU
FIN VER_ARCHIVO
F2
EDITAR EL
ARCHIVO
?
Si
No
RECUPERA LAS
VENTANAS DEL
DIRECTORIO Y MENU
EDITAR
ARCHIVO
CONTENIDO DEL
ARCHIVO AL
EDITOR
FIN VER_ARCHIVO
F3
RENOMBRAR EL
ARCHIVO
?
Si
No
ACTUALIZA LAS
VENTANAS DEL
DIRECTORIO Y MENU
RENOMBRAR
RENOMBRA EL
ARCHIVO
FIN VER_ARCHIVO
F4
MOVER EL
ARCHIVO
?
Si
No
RECUPERA LAS
VENTANAS DEL
DIRECTORIO Y MENU
MOVER
ARCHIVO
MUEVE EL ARCH.
DE UN DIR.
A OTRO
FIN VER_ARCHIVO
F5
BORRAR EL
ARCHIVO
?
Si
No
RECUPERA LAS
VENTANAS DEL
DIRECTORIO Y MENU
BORRAR
ARCHIVO
ELIMINA EL
ARCHIVO
DEL DISCO
FIN VER_ARCHIVO
F7
CAMBIAR DE
DIRECTORIO
?
Si
No
53
54

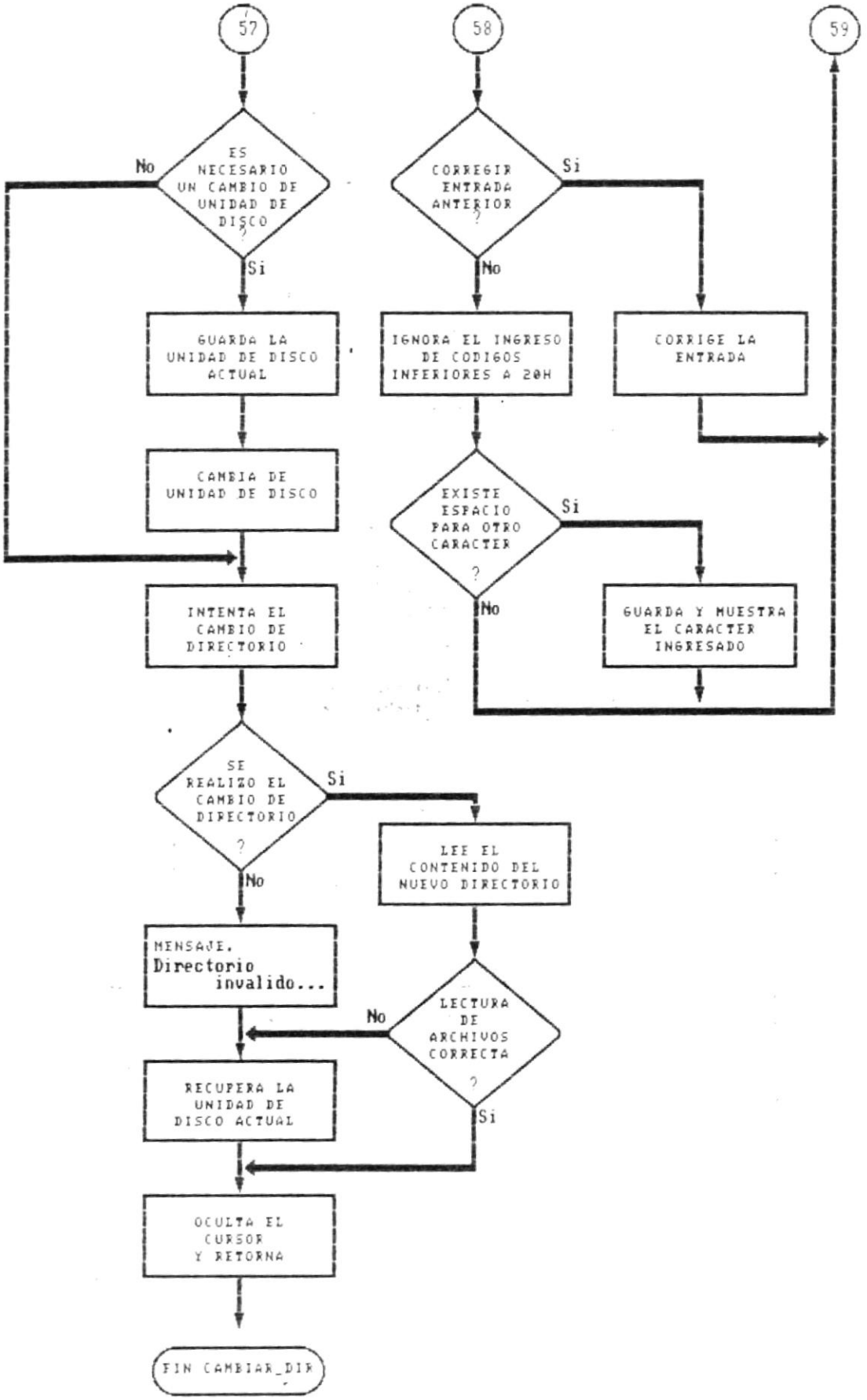

52
53
54
CHEQUEA LAS TECLAS DEL MOV. DEL CURSOR
RECUPERA LAS VENTANAS DEL DIRECTORIO Y MENU
No
DESPLAZAR EL CONTENIDO DE LA PAN TALLA ?
CAMBIAR DE DIRECTORIO
LEE UN NUEVO DIRECTORIO
Si
FIN VER_ARCHIVO
DESPLAZA EL CONTENIDO DEL ARCHIVO
Y LO MUESTRA EN LA PANTALLA 24 LIN. X 80 COL

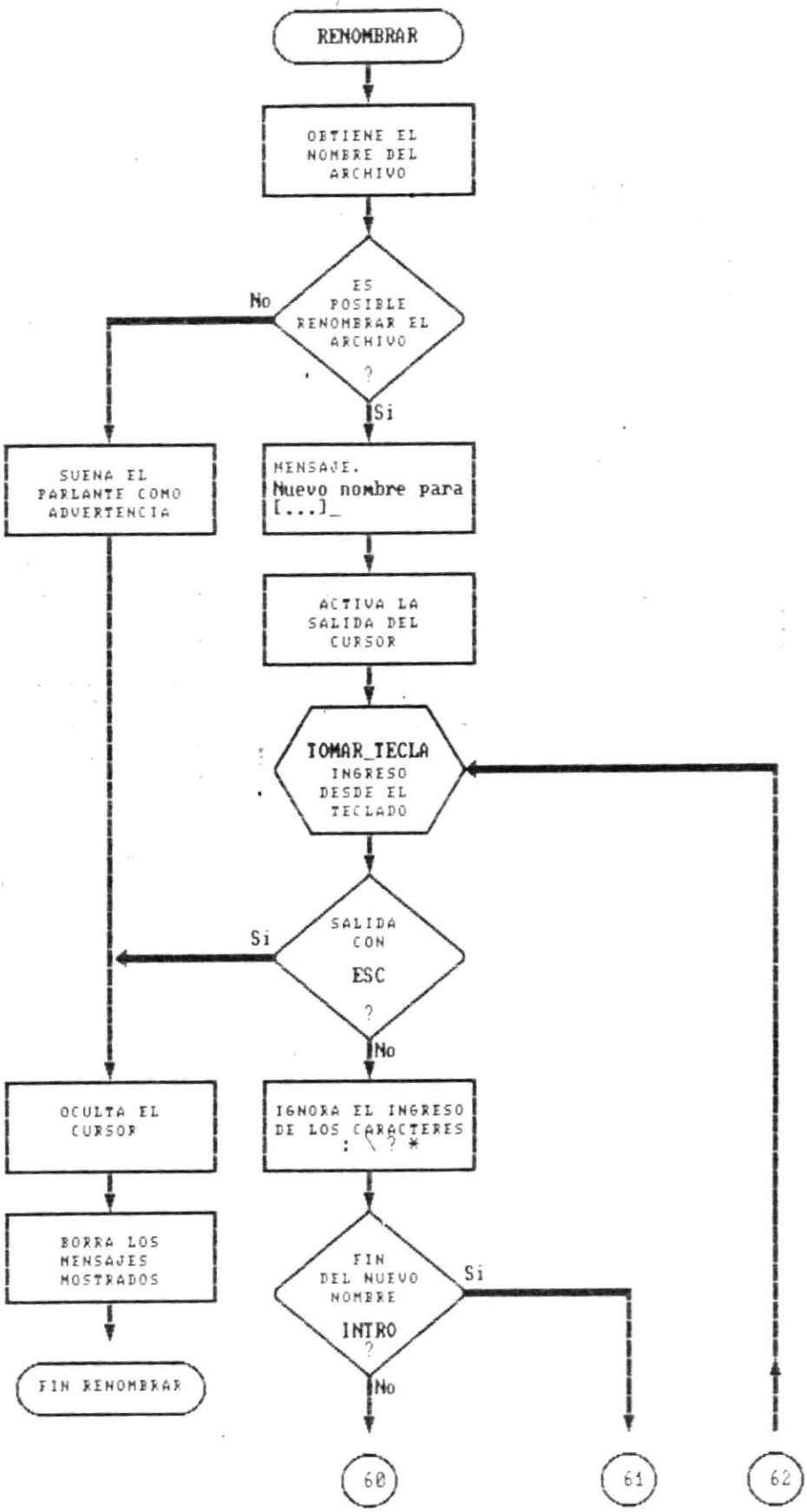

EDITAR_ARCHIVO
TOMA EL NOMBRE DEL ARCHIVO
No
ES POSIBLE EDITAR EL ARCHIVO ?
Si
MENSAJE.
El archivo [...] sera editado Quiere hacerlo.?
SUENA EL PARLANTE PARA INDICAR ERROR
OCULTA EL CURSOR Y SALE
ACTIVA EL CURSOR PARA TOMAR RESPUESTA
FIN EDITAR_ARCH
56

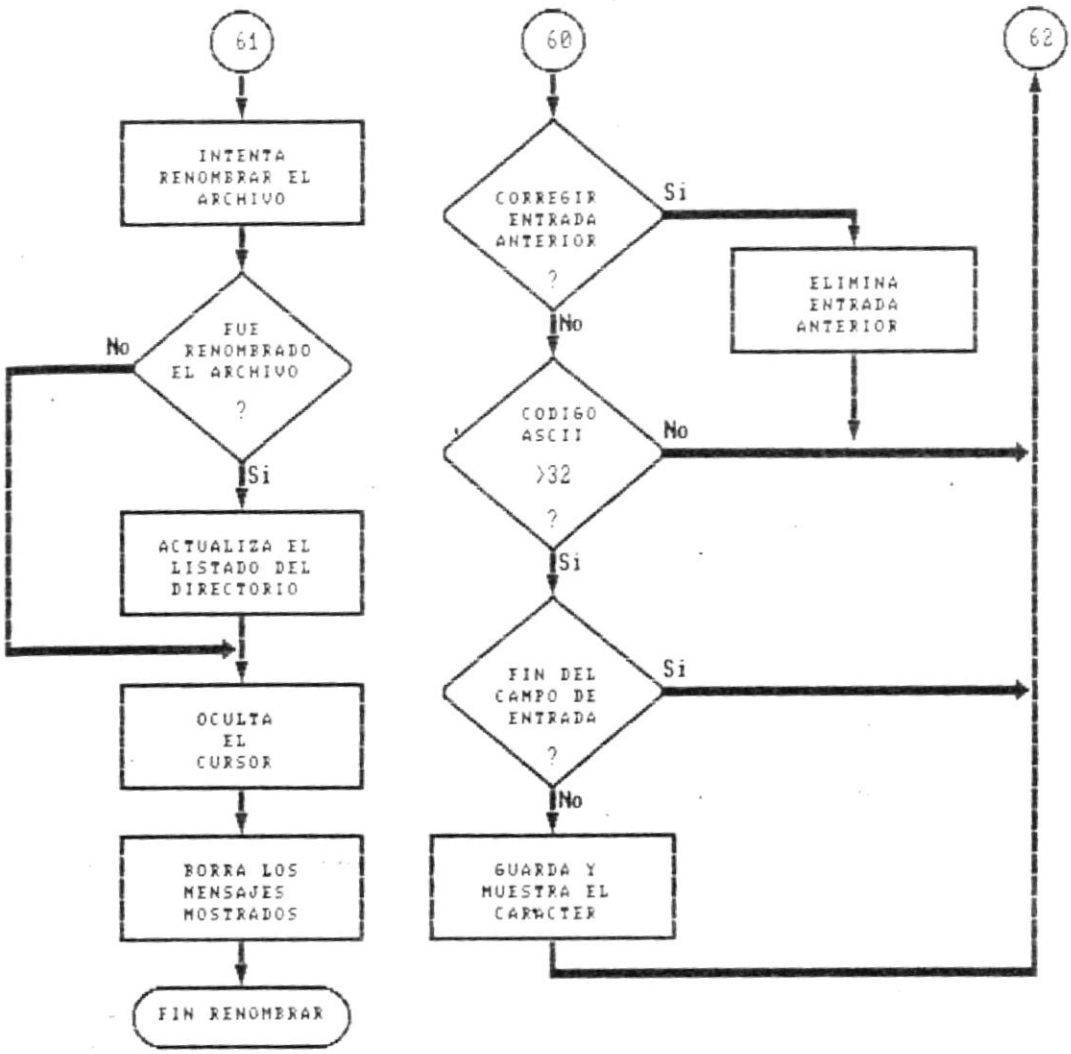

56
SIONO
CHEQUEA
RESPUESTA
DE SI O NO
No
[S]
?
Si
ABRE EL
ARCHIVO PARA
LECTURA
LEE HASTA
7800 BYTES
DEL ARCHIVO
PREPARA EL TEXTO
PARA EL
EDITOR
CIERRA EL
ARCHIVO
BORRA MENSAJES,
OCULTA EL
CURSOR Y SALE
FIN EDITAR_ARCH

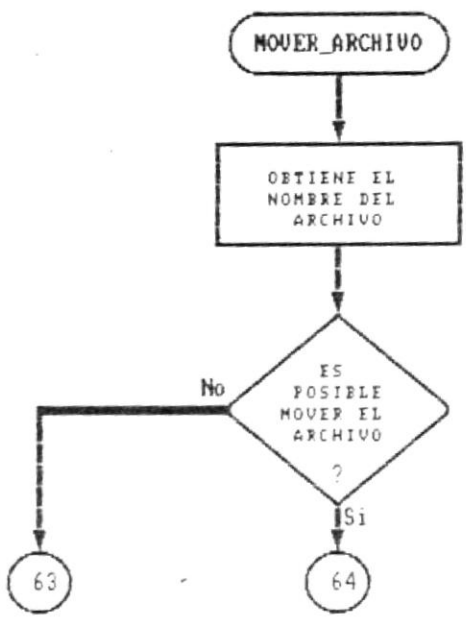

BORRAR_ARCHIVO
TOMA EL NOMBRE DEL ARCHIVO
ESTA PERMITIDO BORRAR EL ARCHIVO ?
No
Si
SUENA EL PARLANTE PARA INDICAR ERROR
MOSTRAR EL MENSAJE DE ADVERTENCIA ?
No
Si
MENSAJE.
[..] sera borrado
Esta seguro ? S/N
ACTIVA LA SALIDA DEL CURSOR
SIONO
ESPERA POR SI O NO
[S] ?
No
Si
BORRA LOS MENSAJES MOSTRADOS
ELIMINA EL EL ARCHIVO DEL DISCO
OCULTA EL CURSOR Y SALE
ELIMINA EL ARCHIVO DEL LISTADO DEL DIR.
FIN BORRAR_ARCH

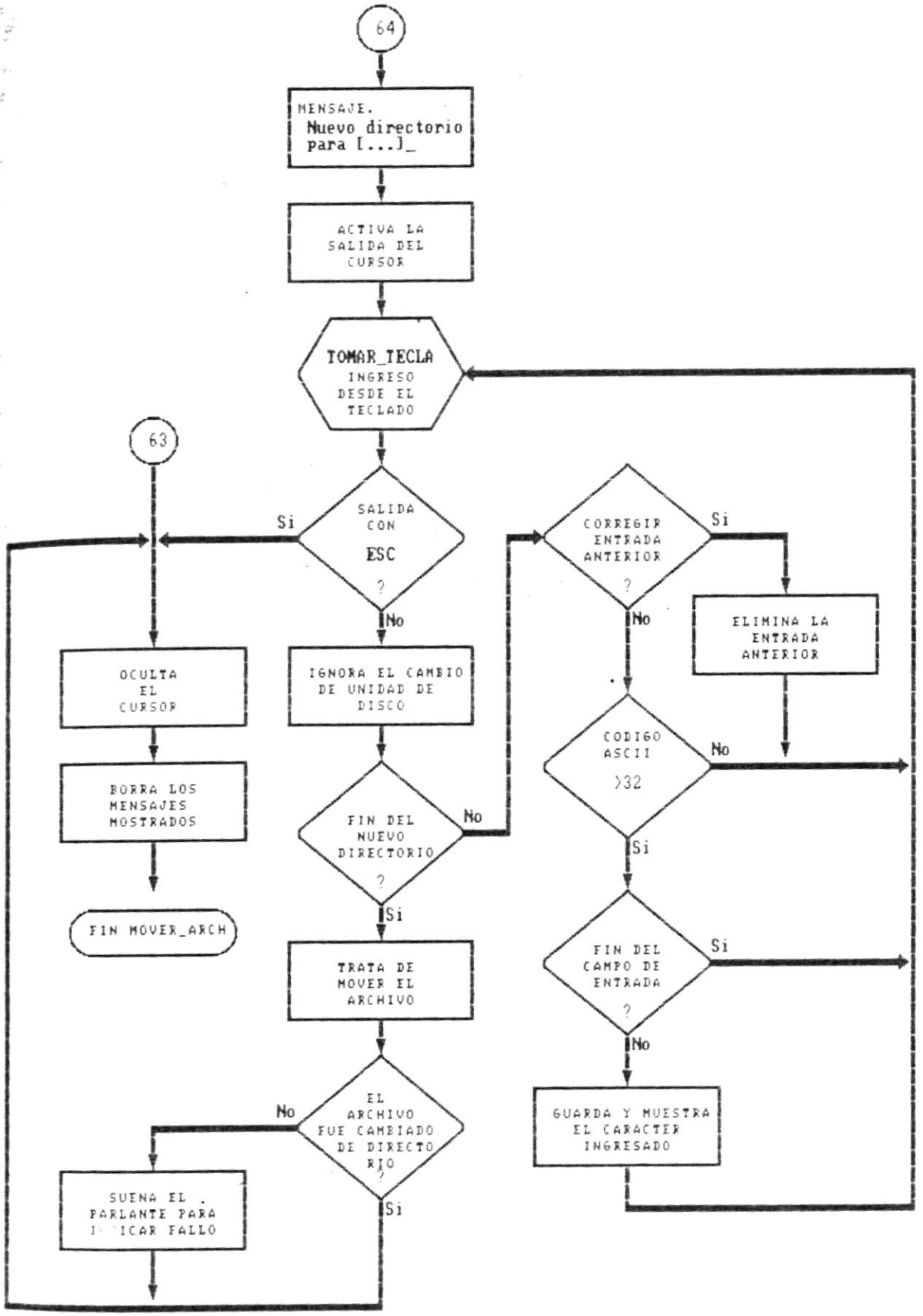

CAMBIAR_DE_DIR
BORRA LOS MENSAJES ANTERIORES
MENSAJE
Cambio de directorio...
ACTIVA LA SALIDA DEL CURSOR
PREPARA AREA PARA NUEVO DIRECTORIO
TOMAR_TECLA
CHEQUEA EL TECLADO
59
SALIDA CON ESC ?
Si
FIN CAMBIO_DIR
No
IGNORA EL INGRESO DE LOS CARACTERES * ?
FIN DEL NUEVO DIR. INTRO ?
No
Si
OCULTA EL CURSOR Y BORRA EL MENSAJE ANTERIOR
57
58

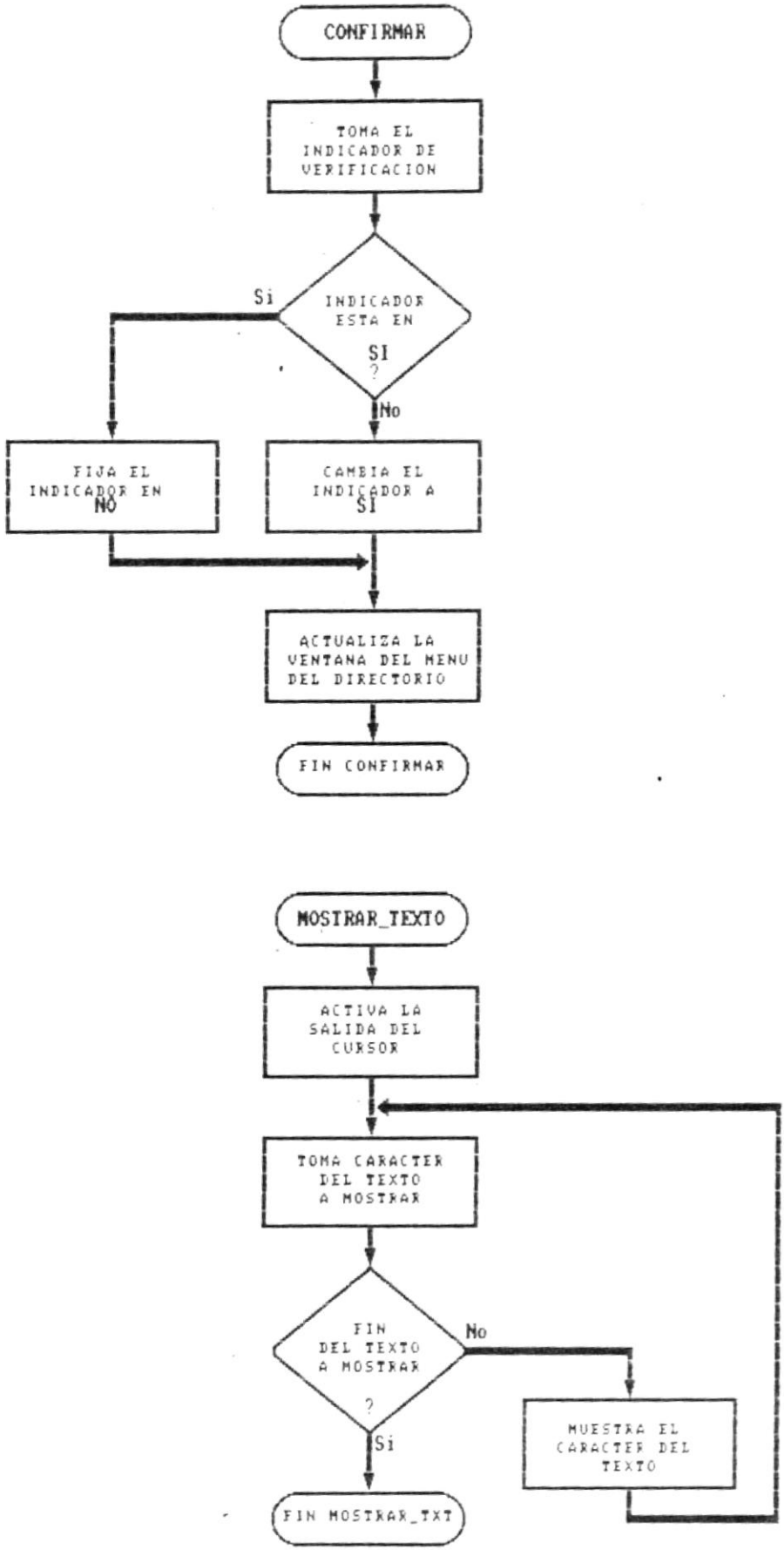

57
58
59
ES NECESARIO UN CAMBIO DE UNIDAD DE DISCO ?
No
Si
GUARDA LA UNIDAD DE DISCO ACTUAL
CAMBIA DE UNIDAD DE DISCO
INTENTA EL CAMBIO DE DIRECTORIO
SE REALIZO EL CAMBIO DE DIRECTORIO ?
Si
No
LEE EL CONTENIDO DEL NUEVO DIRECTORIO
LECTURA DE ARCHIVOS CORRECTA ?
No
Si
MENSAJE, Directorio invalido...
RECUPERA LA UNIDAD DE DISCO ACTUAL
OCULTA EL CURSOR Y RETORNA
FIN CAMBIAR_DIR
CORREGIR ENTRADA ANTERIOR ?
Si
No
IGNORA EL INGRESO DE CODIGOS INFERIORES A 20H
CORRIGE LA ENTRADA
EXISTE ESPACIO PARA OTRO CARACTER ?
Si
No
GUARDA Y MUESTRA EL CARACTER INGRESADO

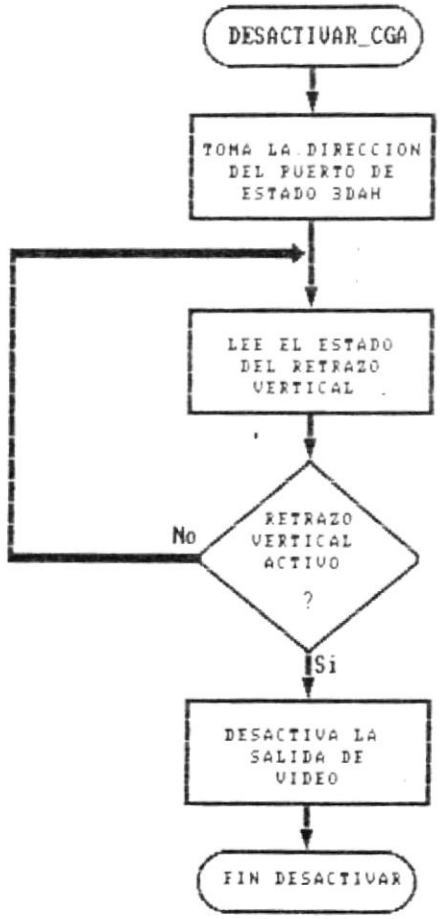

RENOMBRAR
OBTIENE EL NOMBRE DEL ARCHIVO
ES POSIBLE RENOMBRAR EL ARCHIVO ?
No
Si
SUENA EL PARLANTE COMO ADVERTENCIA
MENSAJE. Nuevo nombre para [...]_
ACTIVA LA SALIDA DEL CURSOR
TOMAR_TECLA INGRESO DESDE EL TECLADO
SALIDA CON ESC ?
Si
No
OCULTA EL CURSOR
IGNORA EL INGRESO DE LOS CARACTERES : \ ? *
BORRA LOS MENSAJES MOSTRADOS
FIN DEL NUEVO NOMBRE INTRO ?
Si
No
FIN RENOMBRAR
60
61
62

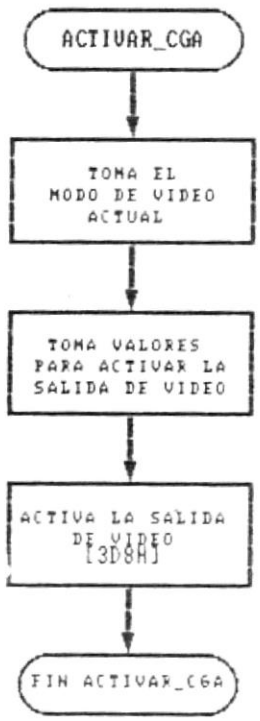

61
INTENTA RENOMBRAR EL ARCHIVO
FUE RENOMBRADO EL ARCHIVO ?
No
Si
ACTUALIZA EL LISTADO DEL DIRECTORIO
OCULTA EL CURSOR
BORRA LOS MENSAJES MOSTRADOS
FIN RENOMBRAR
60
CORREGIR ENTRADA ANTERIOR ?
Si
ELIMINA ENTRADA ANTERIOR
No
CODIGO ASCII >32 ?
No
Si
FIN DEL CAMPO DE ENTRADA ?
Si
No
GUARDA Y MUESTRA EL CARACTER
62

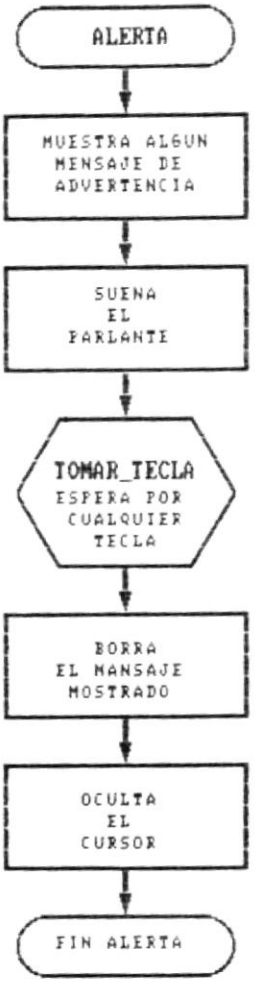

MOVER_ARCHIVO
OBTIENE EL NOMBRE DEL ARCHIVO
ES POSIBLE MOVER EL ARCHIVO ?
No
Si
63
64

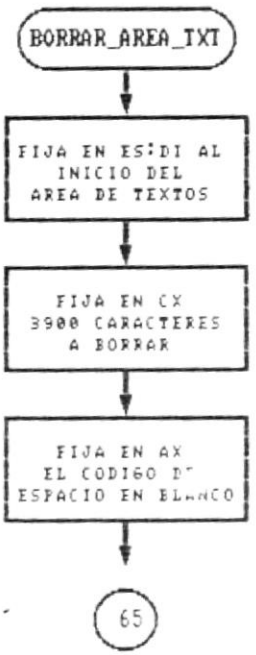

64
MENSAJE.
Nuevo directorio para [...]_
ACTIVA LA SALIDA DEL CURSOR
TOMAR_TECLA
INGRESO DESDE EL TECLADO
63
SALIDA CON ESC ?
Si
No
OCULTA EL CURSOR
BORRA LOS MENSAJES MOSTRADOS
FIN MOVER_ARCH
IGNORA EL CAMBIO DE UNIDAD DE DISCO
FIN DEL NUEVO DIRECTORIO ?
No
Si
TRATA DE MOVER EL ARCHIVO
EL ARCHIVO FUE CAMBIADO DE DIRECTORIO ?
No
Si
SUENA EL PARLANTE PARA INDICAR FALLO
CORREGIR ENTRADA ANTERIOR ?
Si
No
ELIMINA LA ENTRADA ANTERIOR
CODIGO ASCII >32
No
Si
FIN DEL CAMPO DE ENTRADA ?
Si
No
GUARDA Y MUESTRA EL CARACTER INGRESADO

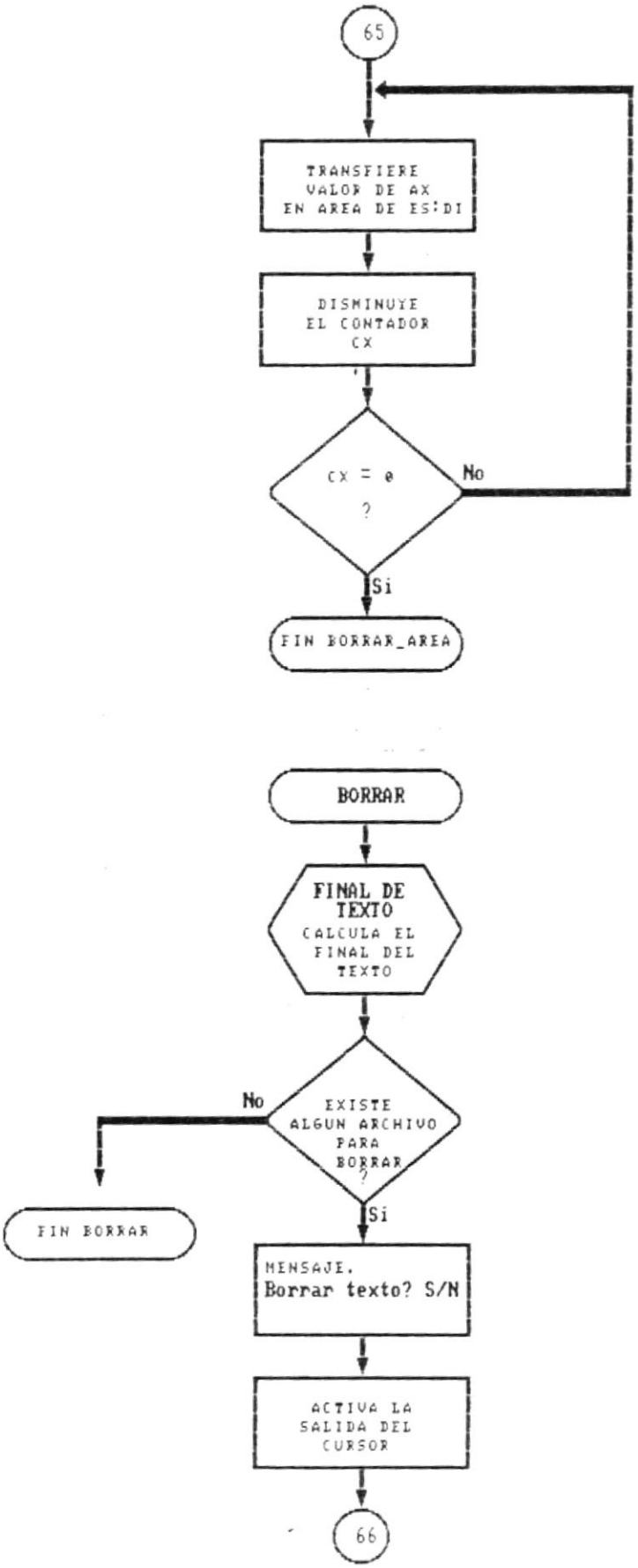

CONFIRMAR
TOMA EL INDICADOR DE VERIFICACION
INDICADOR ESTA EN SI ?
Si
No
FIJA EL INDICADOR EN NO
CAMBIA EL INDICADOR A SI
ACTUALIZA LA VENTANA DEL MENU DEL DIRECTORIO
FIN CONFIRMAR
MOSTRAR_TEXTO
ACTIVA LA SALIDA DEL CURSOR
TOMA CARACTER DEL TEXTO A MOSTRAR
FIN DEL TEXTO A MOSTRAR ?
No
Si
MUESTRA EL CARACTER DEL TEXTO
FIN MOSTRAR_TXT

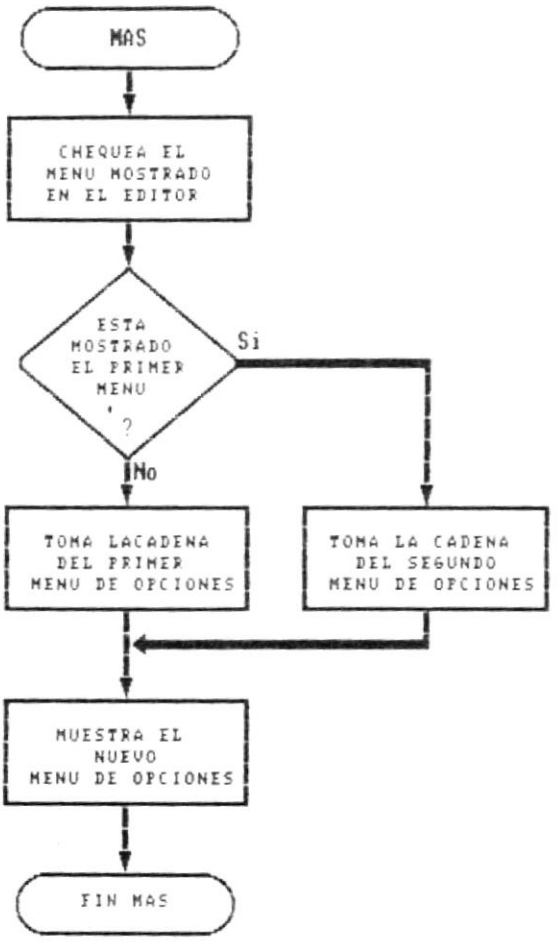

DESACTIVAR_CGA
TOMA LA DIRECCION DEL PUERTO DE ESTADO 3DAH
LEE EL ESTADO DEL RETRAZO VERTICAL
No
RETRAZO VERTICAL ACTIVO
?
Si
DESACTIVA LA SALIDA DE VIDEO
FIN DESACTIVAR

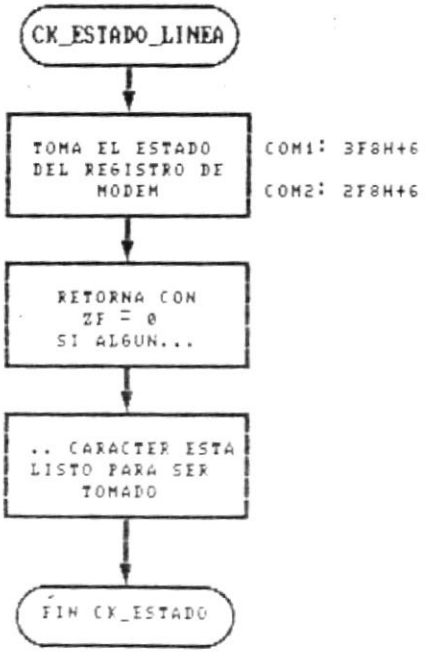

ACTIVAR_CGA
TOMA EL MODO DE VIDEO ACTUAL
TOMA VALORES PARA ACTIVAR LA SALIDA DE VIDEO
ACTIVA LA SALIDA DE VIDEO
[3D8H]
FIN ACTIVAR_CGA

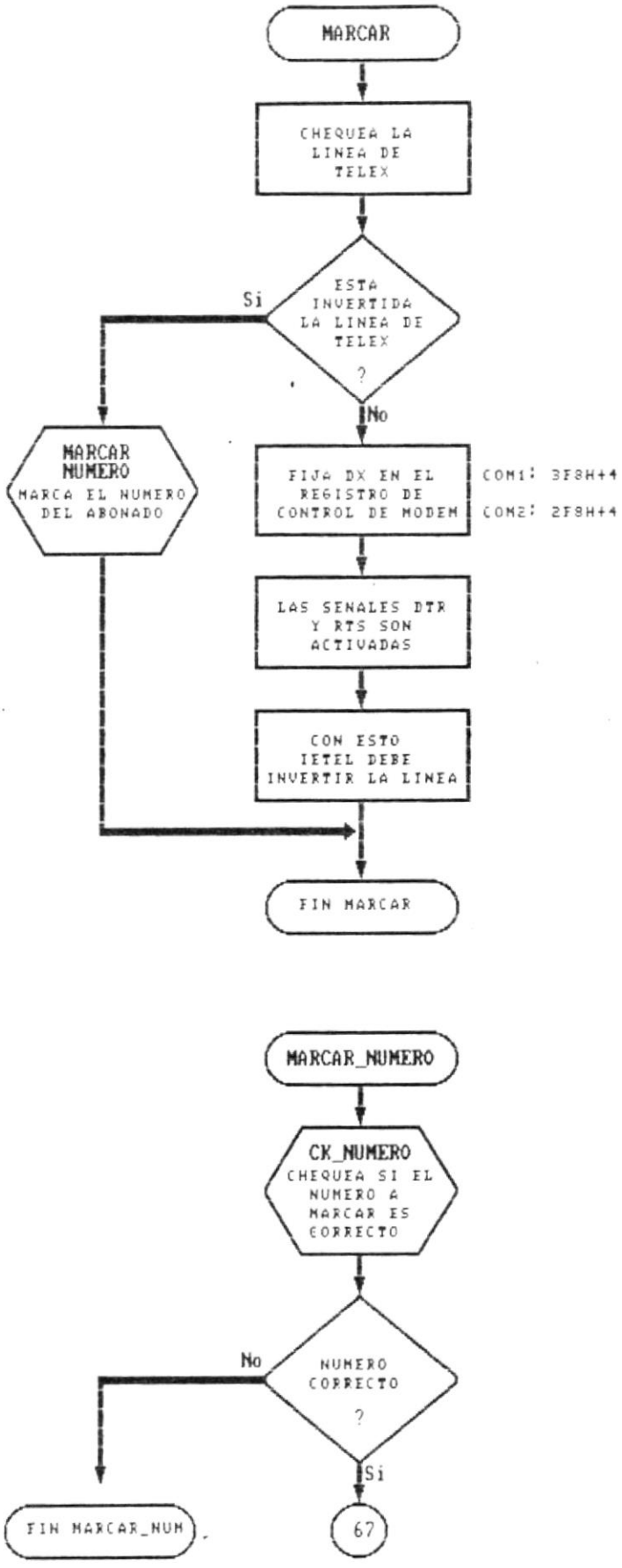

ALERTA
MUESTRA ALGUN MENSAJE DE ADVERTENCIA
SUENA EL PARLANTE
TOMAR_TECLA
ESPERA POR CUALQUIER TECLA
BORRA EL MANSAJE MOSTRADO
OCULTA EL CURSOR
FIN ALERTA

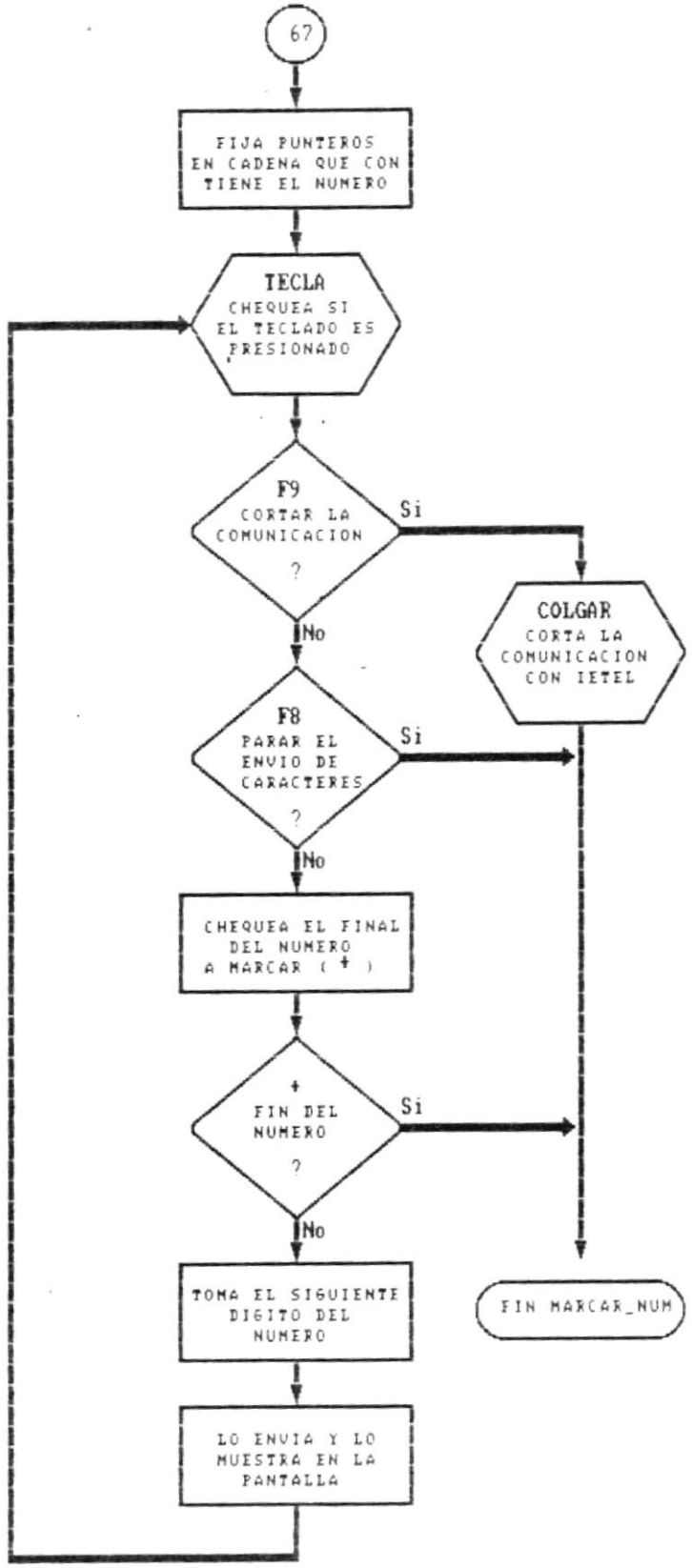

BORRAR_AREA_TXT
FIJA EN ES:DI AL INICIO DEL AREA DE TEXTOS
FIJA EN CX 3900 CARACTERES A BORRAR
FIJA EN AX EL CODIGO D
ESPACIO EN BLANCO

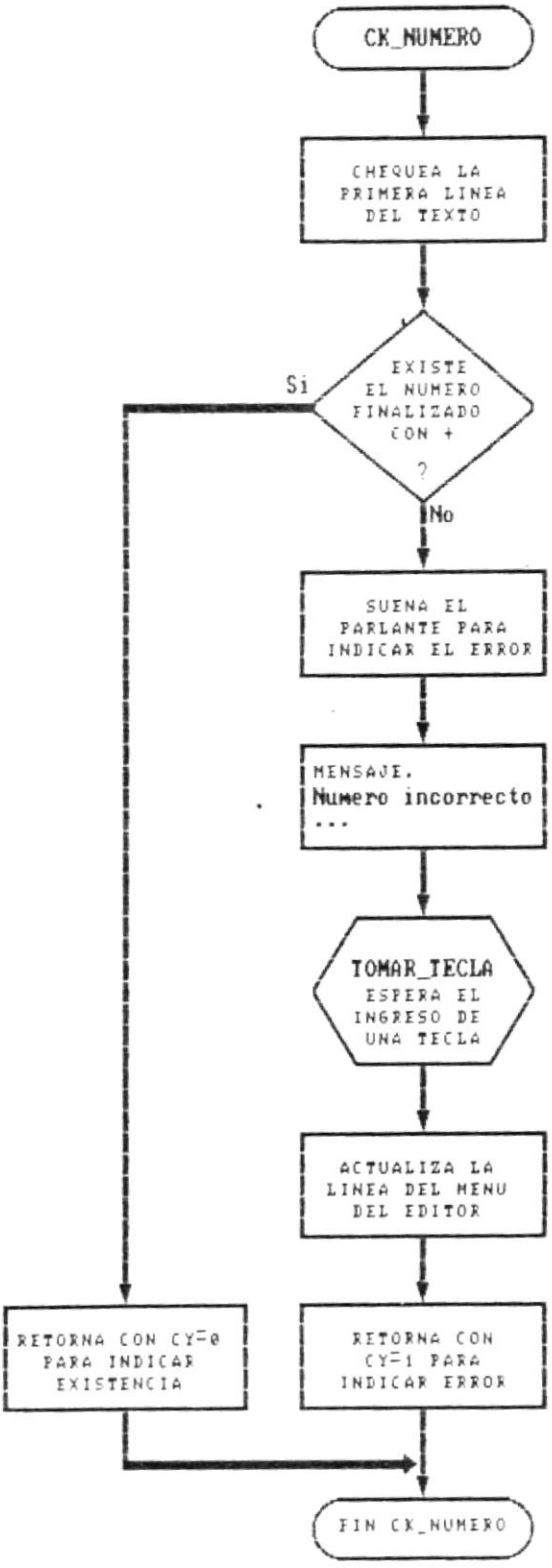

65
TRANSFIERE VALOR DE AX EN AREA DE ES:DI
DISMINUYE EL CONTADOR CX
CX = 0 ?
No
Si
FIN BORRAR_AREA
BORRAR
FINAL DE TEXTO
CALCULA EL FINAL DEL TEXTO
EXISTE ALGUN ARCHIVO PARA BORRAR ?
No
Si
FIN BORRAR
MENSAJE.
Borrar texto? S/N
ACTIVA LA SALIDA DEL CURSOR
66

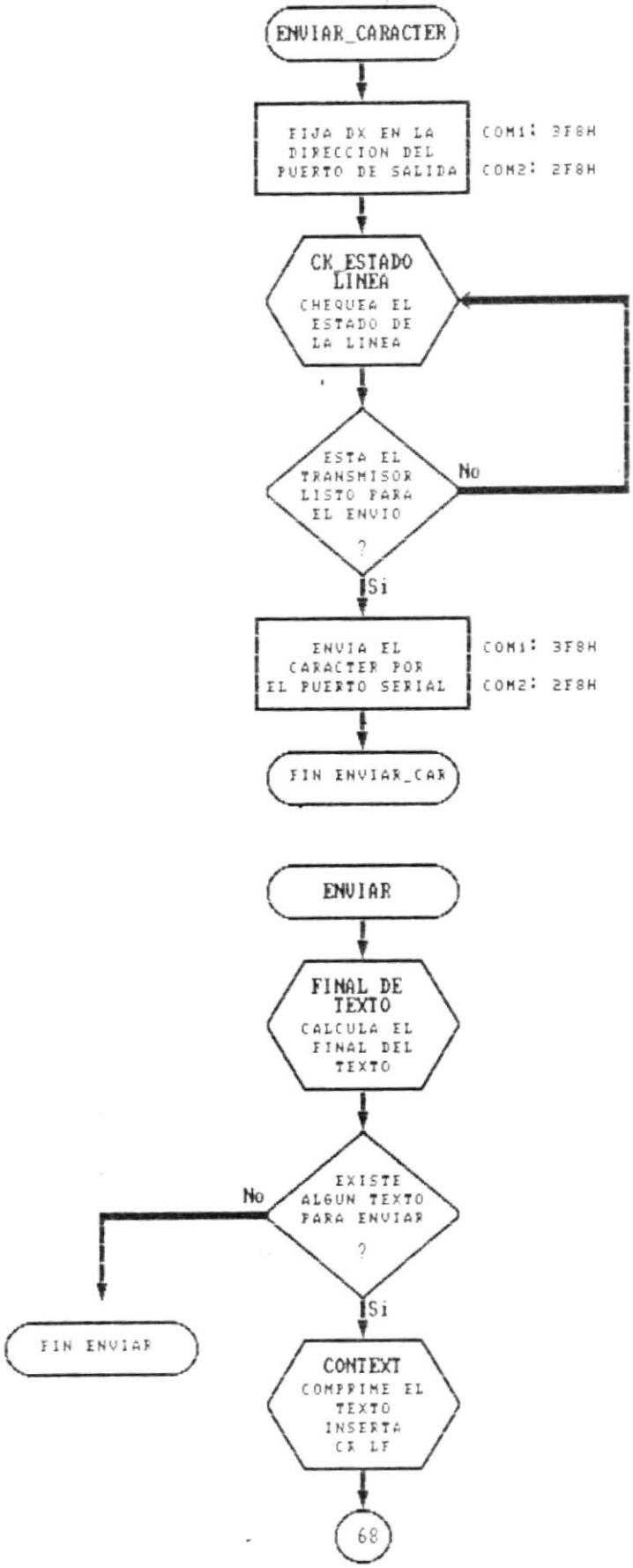

MAS
CHEQUEA EL MENU MOSTRADO EN EL EDITOR
ESTA MOSTRADO EL PRIMER MENU ?
Si
No
TOMA LACADENA DEL PRIMER MENU DE OPCIONES
TOMA LA CADENA DEL SEGUNDO MENU DE OPCIONES
MUESTRA EL NUEVO MENU DE OPCIONES
FIN MAS

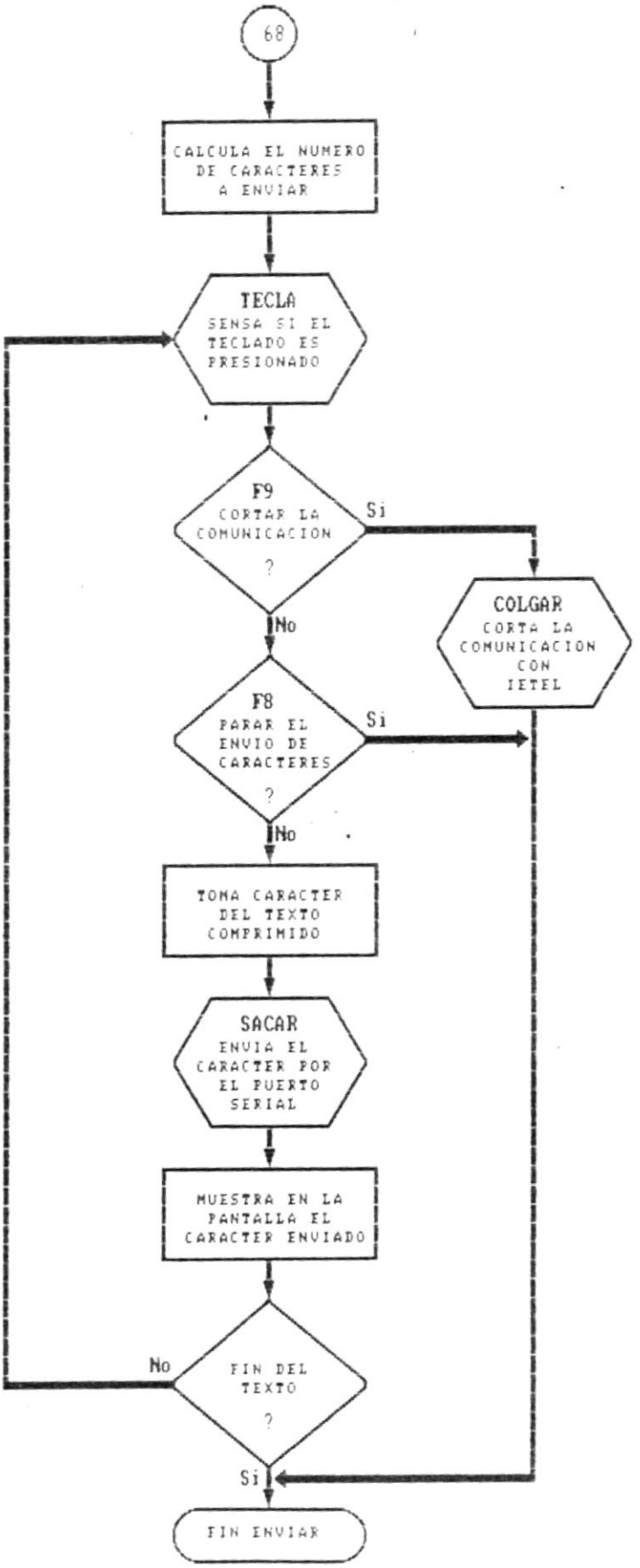

CK_ESTADO_LINEA
TOMA EL ESTADO DEL REGISTRO DE MODEM
COM1: 3F8H+6
COM2: 2F8H+6
RETORNA CON ZF = 0 SI ALGUN...
.. CARACTER ESTA LISTO PARA SER TOMADO
FIN CK_ESTADO

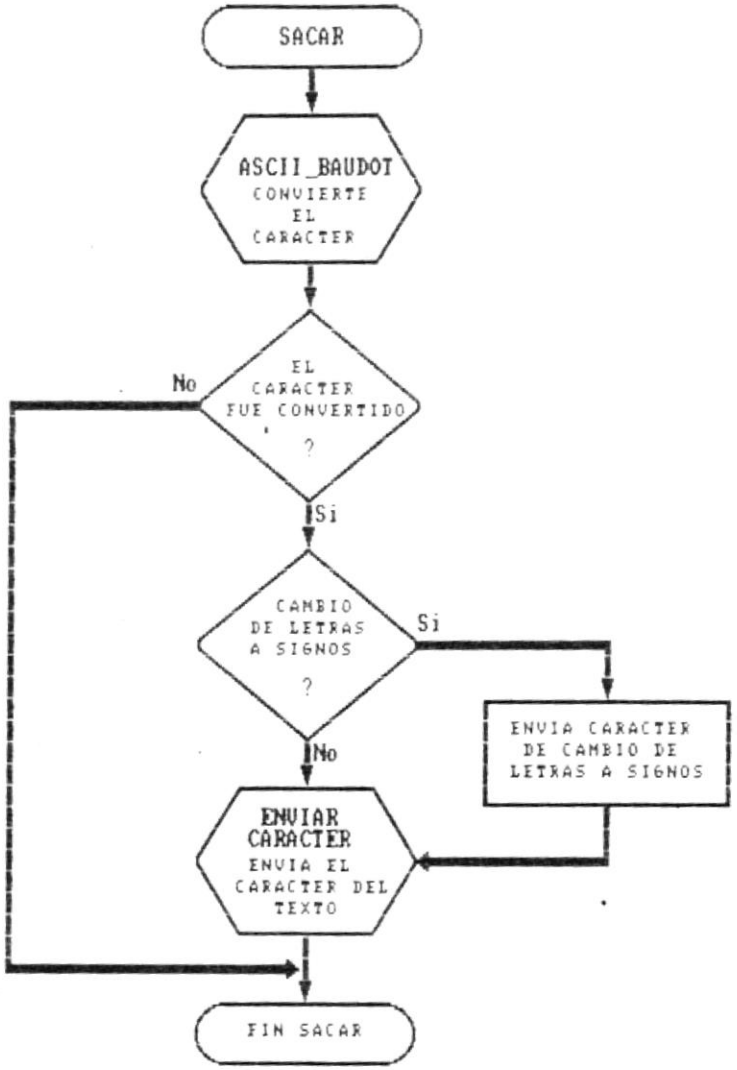

MARCAR
CHEQUEA LA LINEA DE TELEX
ESTA INVERTIDA LA LINEA DE TELEX ?
Si
No
MARCAR NUMERO
MARCA EL NUMERO DEL ABONADO
FIJA DX EN EL REGISTRO DE CONTROL DE MODEM
COM1: 3F8H+4
COM2: 2F8H+4
LAS SENALES DTR Y RTS SON ACTIVADAS
CON ESTO IETEL DEBE INVERTIR LA LINEA
FIN MARCAR
MARCAR_NUMERO
CK_NUMERO
CHEQUEA SI EL NUMERO A MARCAR ES CORRECTO
NUMERO CORRECTO ?
No
Si
FIN MARCAR_NUM
67

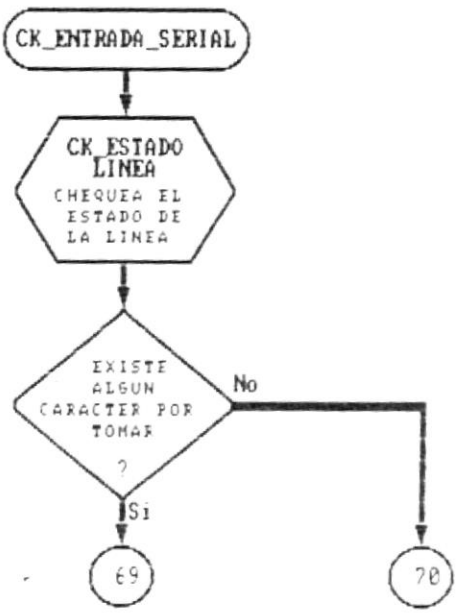

67
FIJA PUNTEROS EN CADENA QUE CONTIENE EL NUMERO
TECLA
CHEQUEA SI EL TECLADO ES PRESIONADO
F9
CORTAR LA COMUNICACION
?
Si
No
COLGAR
CORTA LA COMUNICACION CON IETEL
F8
PARAR EL ENVIO DE CARACTERES
?
Si
No
CHEQUEA EL FINAL DEL NUMERO A MARCAR (+)
+
FIN DEL NUMERO
?
Si
No
TOMA EL SIGUIENTE DIGITO DEL NUMERO
LO ENVIA Y LO MUESTRA EN LA PANTALLA
FIN MARCAR_NUM

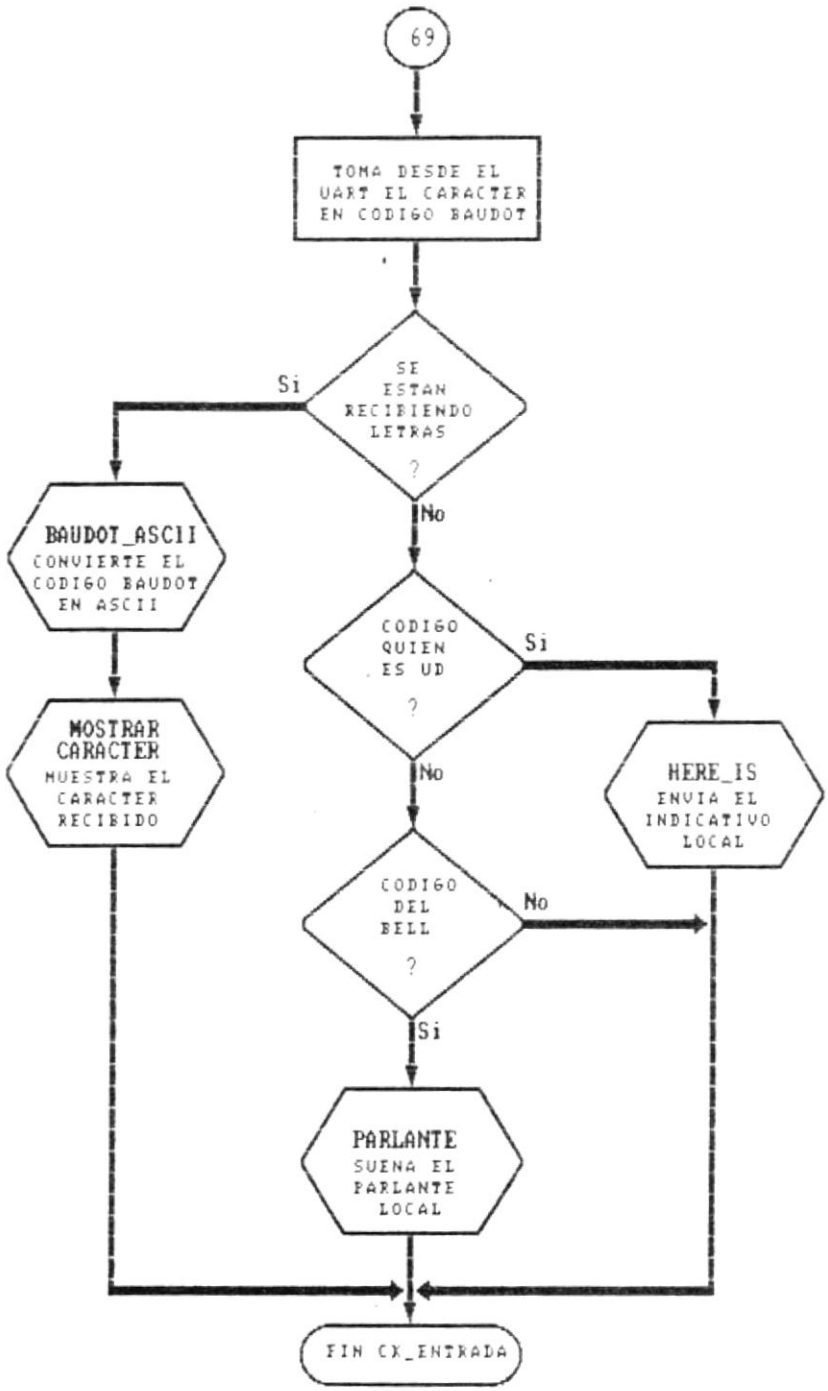

CK_NUMERO
CHEQUEA LA PRIMERA LINEA DEL TEXTO
EXISTE EL NUMERO FINALIZADO CON + ?
Si
No
SUENA EL PARLANTE PARA INDICAR EL ERROR
MENSAJE. Numero incorrecto ...
TOMAR_TECLA ESPERA EL INGRESO DE UNA TECLA
ACTUALIZA LA LINEA DEL MENU DEL EDITOR
RETORNA CON CY=0 PARA INDICAR EXISTENCIA
RETORNA CON CY=1 PARA INDICAR ERROR
FIN CK_NUMERO

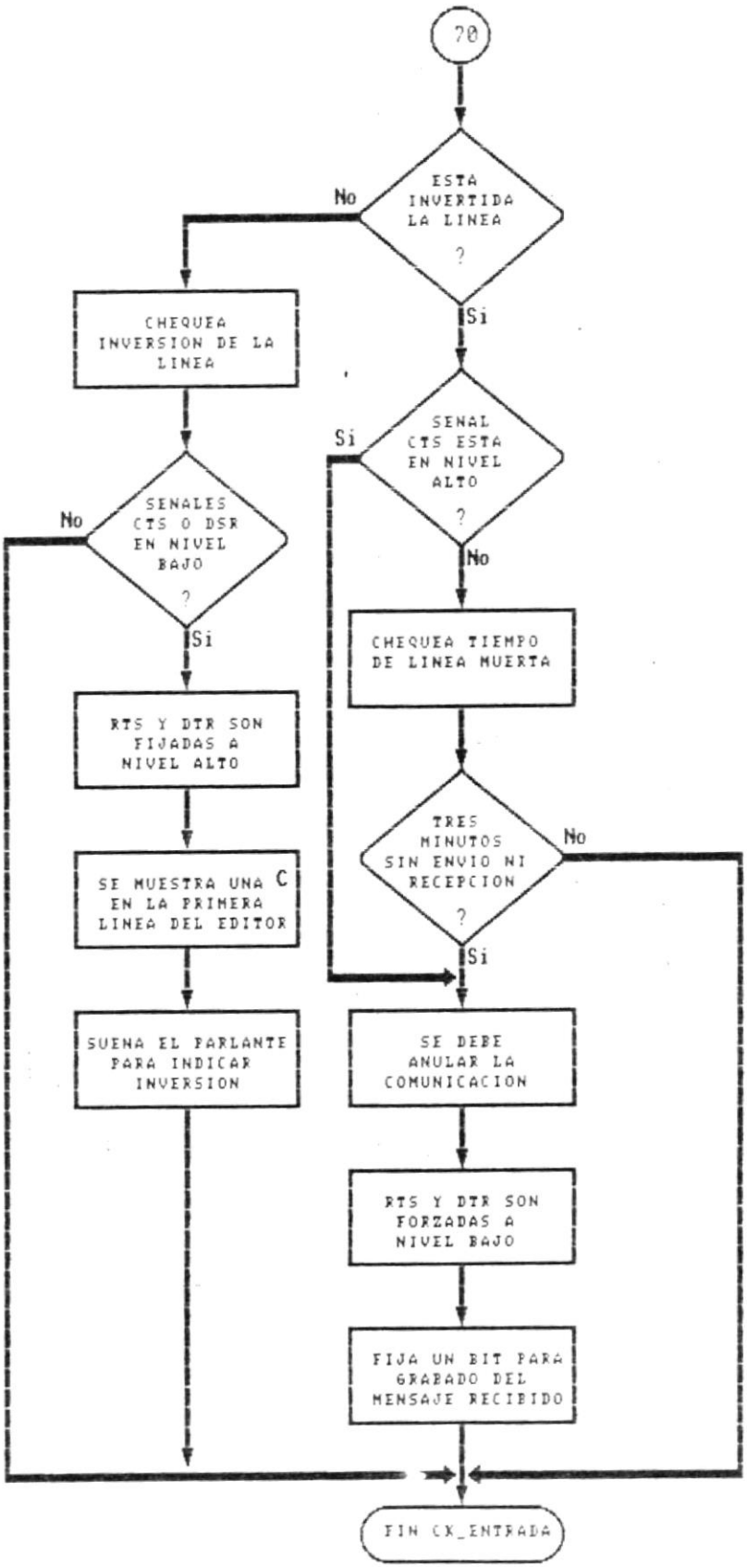

ENVIAR_CARACTER
FIJA DX EN LA DIRECCION DEL PUERTO DE SALIDA
COM1: 3F8H
COM2: 2F8H
CK_ESTADO LINEA
CHEQUEA EL ESTADO DE LA LINEA
ESTA EL TRANSMISOR LISTO PARA EL ENVIO ?
No
Si
ENVIA EL CARACTER POR EL PUERTO SERIAL
COM1: 3F8H
COM2: 2F8H
FIN ENVIAR_CAR
ENVIAR
FINAL DE TEXTO
CALCULA EL FINAL DEL TEXTO
EXISTE ALGUN TEXTO PARA ENVIAR ?
No
Si
FIN ENVIAR
CONTEXT
COMPRIME EL TEXTO
INSERTA CR LF
68

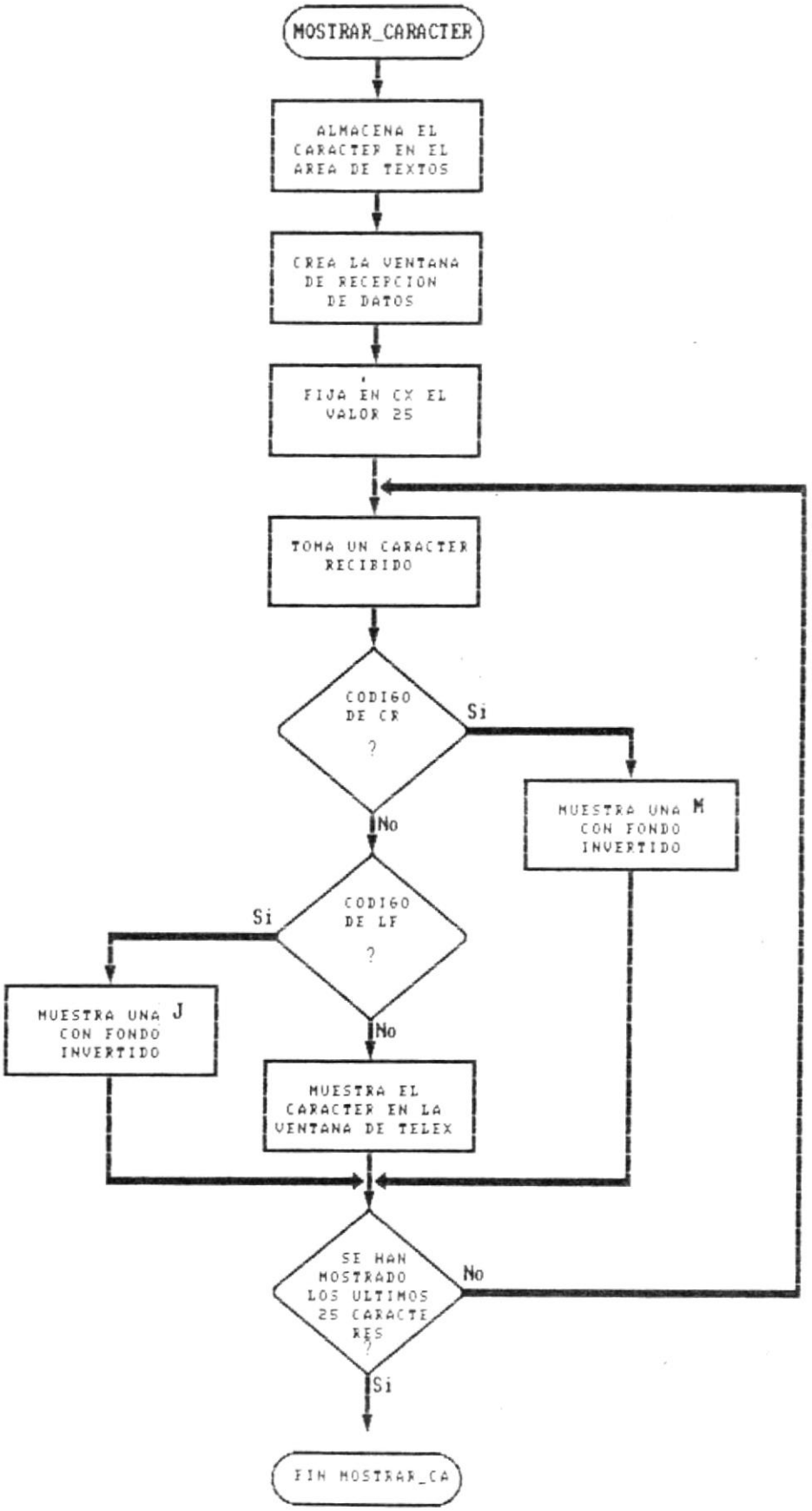

CALCULA EL NUMERO DE CARACTERES A ENVIAR
TECLA
SENSA SI EL TECLADO ES PRESIONADO
F9
CORTAR LA COMUNICACION
?
Si
COLGAR
CORTA LA COMUNICACION CON IETEL
No
F8
PARAR EL ENVIO DE CARACTERES
?
Si
No
TOMA CARACTER DEL TEXTO COMPRIMIDO
SACAR
ENVIA EL CARACTER POR EL PUERTO SERIAL
MUESTRA EN LA PANTALLA EL CARACTER ENVIADO
FIN DEL TEXTO
?
No
Si
FIN ENVIAR

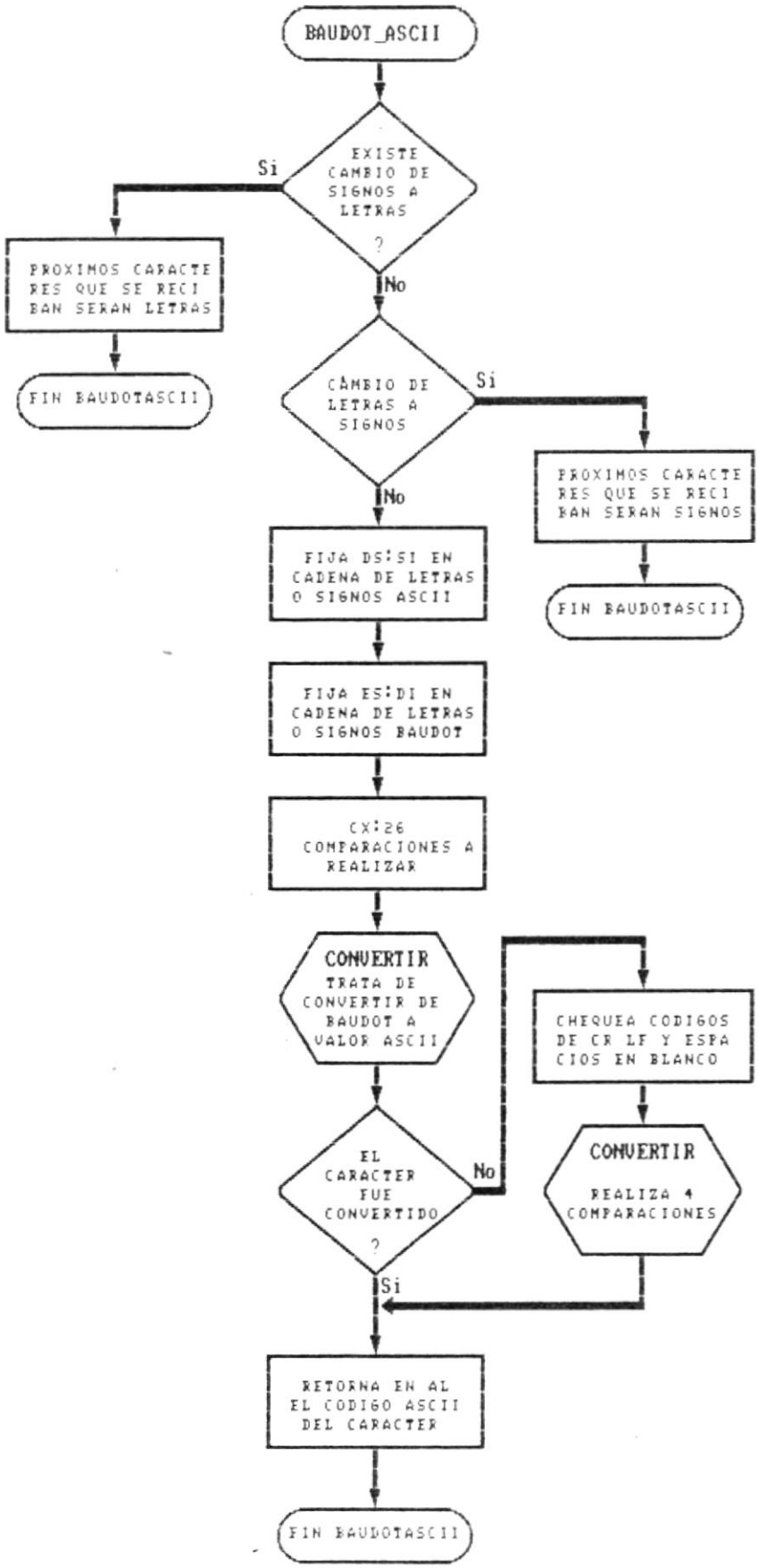

SACAR
ASCII_BAUDOT
CONVIERTE
EL
CARACTER
EL
CARACTER
FUE CONVERTIDO
?
No
Si
CAMBIO
DE LETRAS
A SIGNOS
?
Si
ENVIA CARACTER
DE CAMBIO DE
LETRAS A SIGNOS
No
ENVIAR
CARACTER
ENVIA EL
CARACTER DEL
TEXTO
FIN SACAR

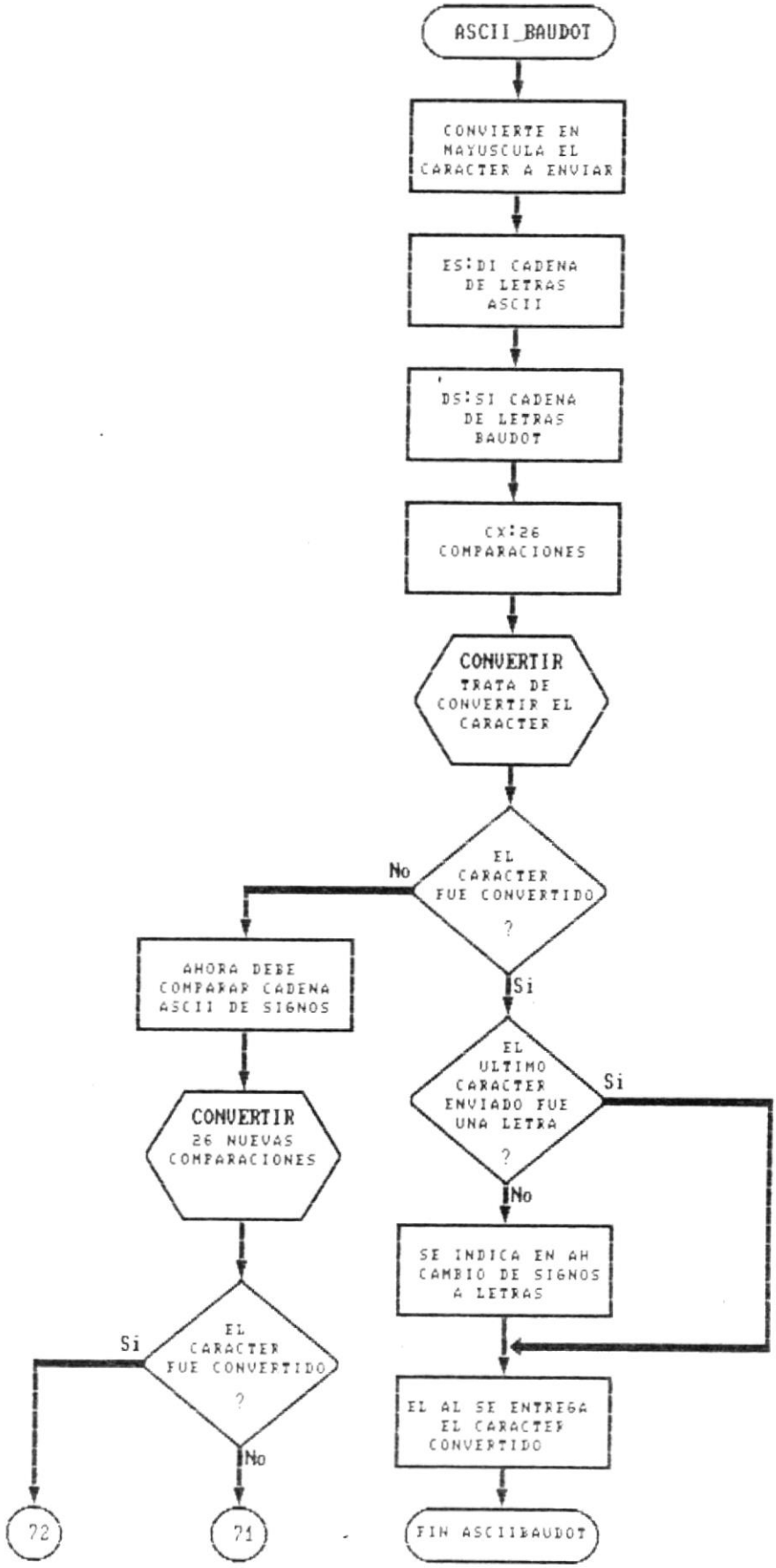

CK_ENTRADA_SERIAL
CK_ESTADO
LINEA
CHEQUEA EL
ESTADO DE
LA LINEA
EXISTE
ALGUN
CARACTER POR
TOMAR
?
No
Si
69
70

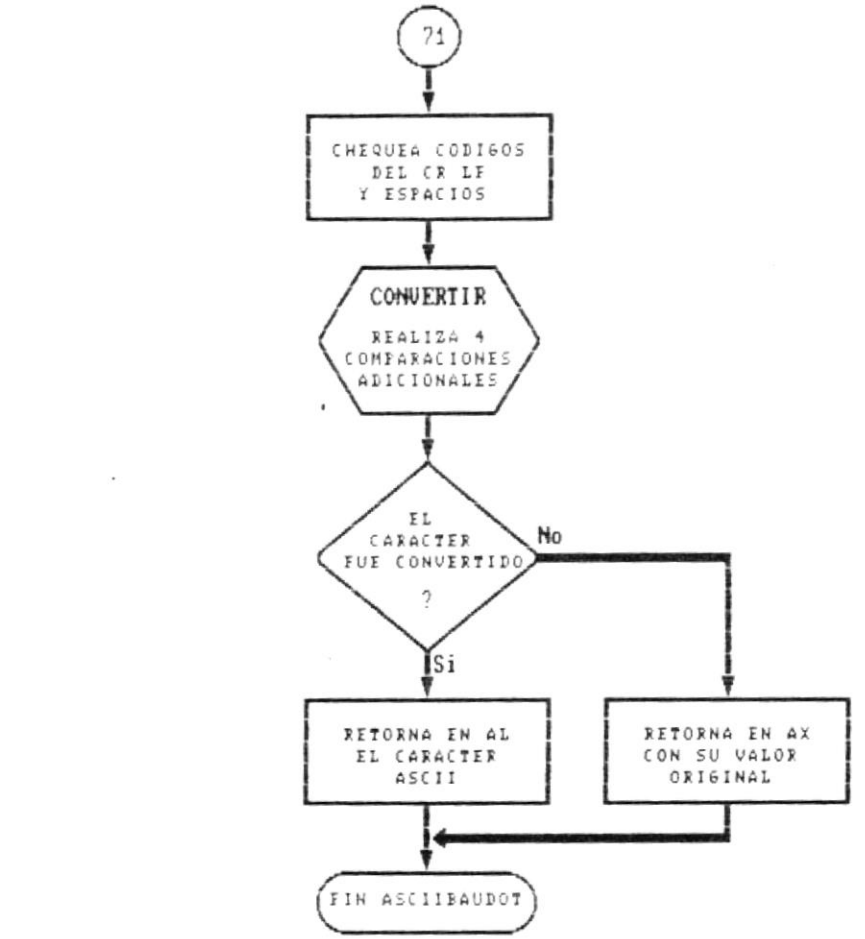

69
TOMA DESDE EL UART EL CARACTER EN CODIGO BAUDOT
SE ESTAN RECIBIENDO LETRAS ?
Si
No
BAUDOT_ASCII
CONVIERTE EL CODIGO BAUDOT EN ASCII
MOSTRAR CARACTER
MUESTRA EL CARACTER RECIBIDO
CODIGO QUIEN ES UD ?
Si
No
HERE_IS
ENVIA EL INDICATIVO LOCAL
CODIGO DEL BELL ?
No
Si
PARLANTE
SUENA EL PARLANTE LOCAL
FIN CX_ENTRADA

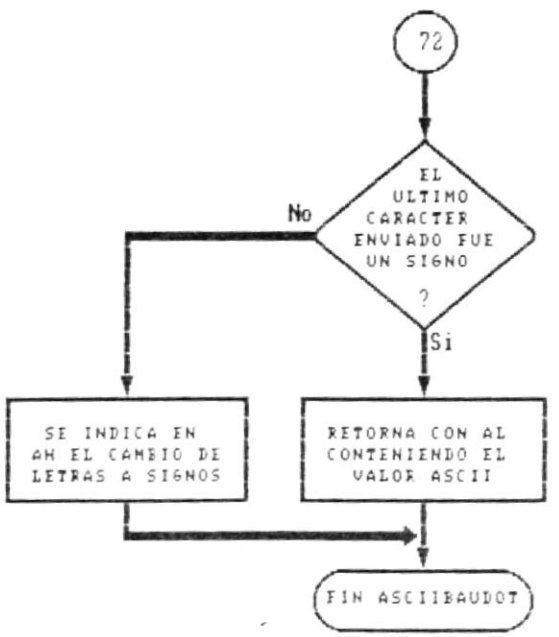

70
ESTA INVERTIDA LA LINEA ?
No
Si
CHEQUEA INVERSION DE LA LINEA
SENAL CTS ESTA EN NIVEL ALTO ?
Si
No
SENALES CTS O DSR EN NIVEL BAJO ?
No
Si
CHEQUEA TIEMPO DE LINEA MUERTA
RTS Y DTR SON FIJADAS A NIVEL ALTO
TRES MINUTOS SIN ENVIO NI RECEPCION ?
No
Si
SE MUESTRA UNA C EN LA PRIMERA LINEA DEL EDITOR
SUENA EL PARLANTE PARA INDICAR INVERSION
SE DEBE ANULAR LA COMUNICACION
RTS Y DTR SON FORZADAS A NIVEL BAJO
FIJA UN BIT PARA GRABADO DEL MENSAJE RECIBIDO
FIN CX_ENTRADA

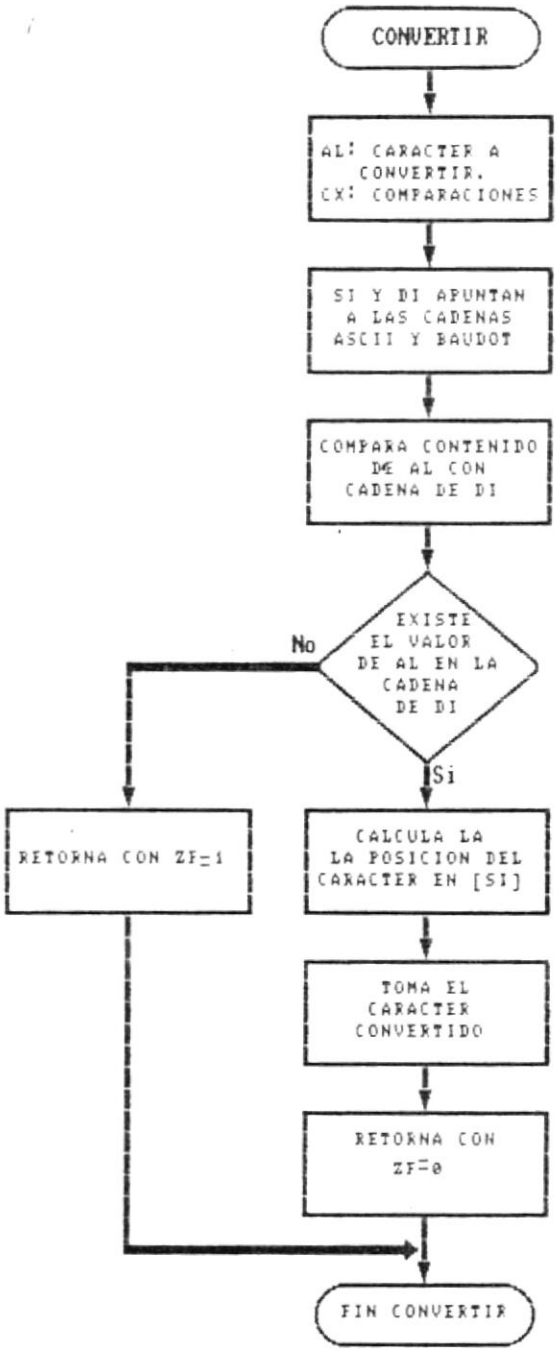

MOSTRAR_CARACTER
ALMACENA EL CARACTER EN EL AREA DE TEXTOS
CREA LA VENTANA DE RECEPCION DE DATOS
FIJA EN CX EL VALOR 25
TOMA UN CARACTER RECIBIDO
CODIGO DE CR ?
Si
MUESTRA UNA M CON FONDO INVERTIDO
No
CODIGO DE LF ?
Si
MUESTRA UNA J CON FONDO INVERTIDO
No
MUESTRA EL CARACTER EN LA VENTANA DE TELEX
SE HAN MOSTRADO LOS ULTIMOS 25 CARACTERES ?
No
Si
FIN MOSTRAR_CA

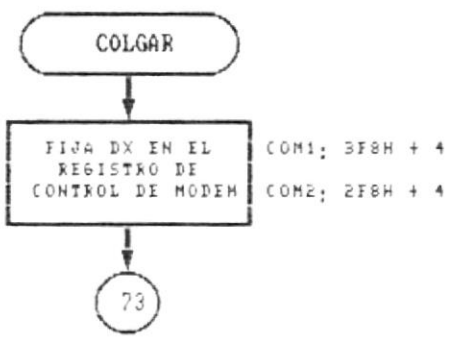

BAUDOT_ASCII
EXISTE CAMBIO DE SIGNOS A LETRAS ?
Si
No
PROXIMOS CARACTERES QUE SE RECIBAN SERAN LETRAS
FIN BAUDOTASCII
CAMBIO DE LETRAS A SIGNOS
Si
No
PROXIMOS CARACTERES QUE SE RECIBAN SERAN SIGNOS
FIN BAUDOTASCII
FIJA DS:SI EN CADENA DE LETRAS O SIGNOS ASCII
FIJA ES:DI EN CADENA DE LETRAS O SIGNOS BAUDOT
CX:26 COMPARACIONES A REALIZAR
CONVERTIR
TRATA DE CONVERTIR DE BAUDOT A VALOR ASCII
EL CARACTER FUE CONVERTIDO ?
No
Si
CHEQUEA CODIGOS DE CR LF Y ESPACIOS EN BLANCO
CONVERTIR
REALIZA 4 COMPARACIONES
RETORNA EN AL EL CODIGO ASCII DEL CARACTER
FIN BAUDOTASCII

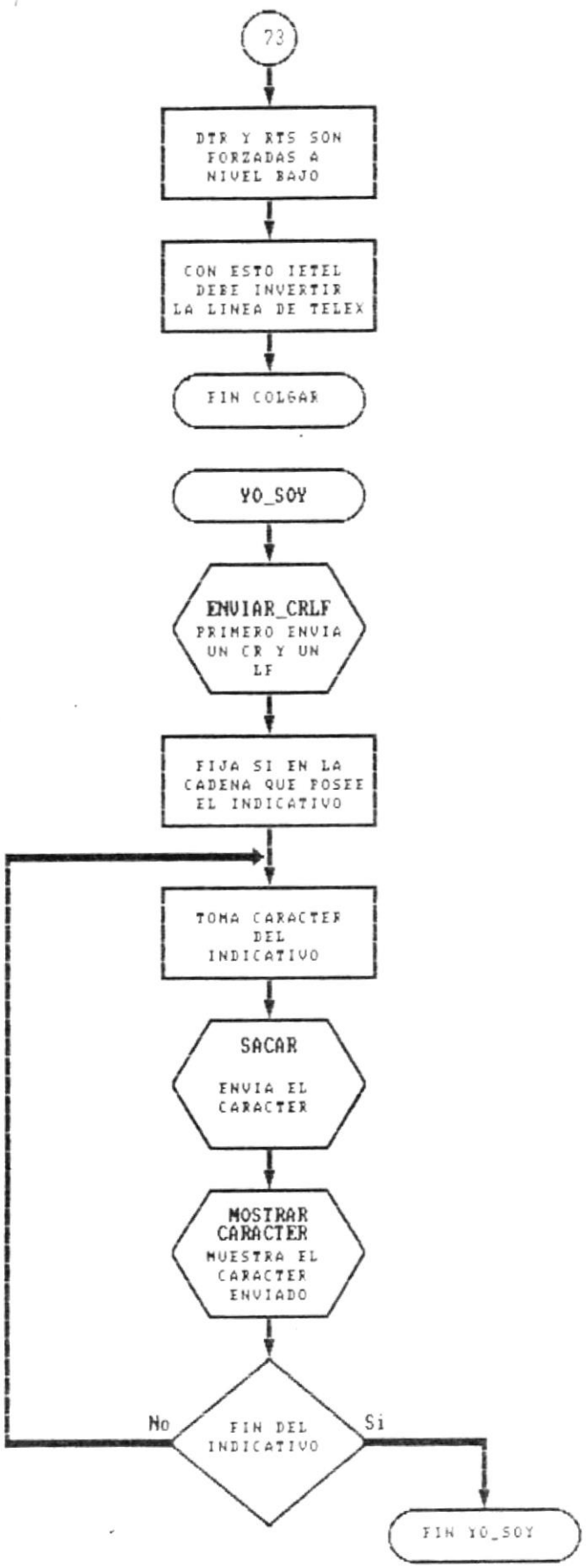

ASCII_BAUDOT
CONVIERTE EN MAYUSCULA EL CARACTER A ENVIAR
ES:DI CADENA DE LETRAS ASCII
DS:SI CADENA DE LETRAS BAUDOT
CX:26 COMPARACIONES
CONVERTIR
TRATA DE CONVERTIR EL CARACTER
EL CARACTER FUE CONVERTIDO ?
No
Si
AHORA DEBE COMPARAR CADENA ASCII DE SIGNOS
CONVERTIR
26 NUEVAS COMPARACIONES
EL CARACTER FUE CONVERTIDO ?
Si
No
72
71
EL ULTIMO CARACTER ENVIADO FUE UNA LETRA ?
Si
No
SE INDICA EN AH CAMBIO DE SIGNOS A LETRAS
EL AL SE ENTREGA EL CARACTER CONVERTIDO
FIN ASCIIBAUDOT


71
CHEQUEA CODIGOS DEL CR LF Y ESPACIOS
CONVERTIR
REALIZA 4 COMPARACIONES ADICIONALES
EL CARACTER FUE CONVERTIDO ?
No
Si
RETORNA EN AL EL CARACTER ASCII
RETORNA EN AX CON SU VALOR ORIGINAL
FIN ASCIIBAUDOT

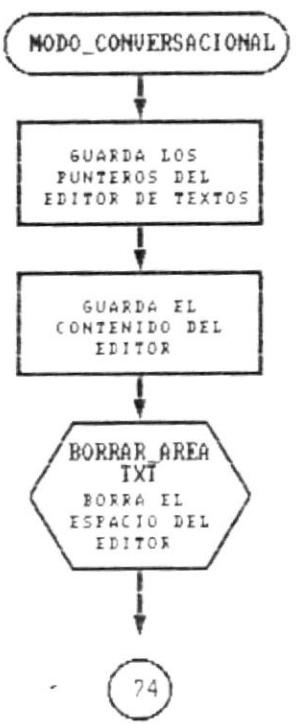

72
EL ULTIMO CARACTER ENVIADO FUE UN SIGNO ?
No
Si
SE INDICA EN AH EL CAMBIO DE LETRAS A SIGNOS
RETORNA CON AL CONTENIENDO EL VALOR ASCII
FIN ASCIIBAUDOT

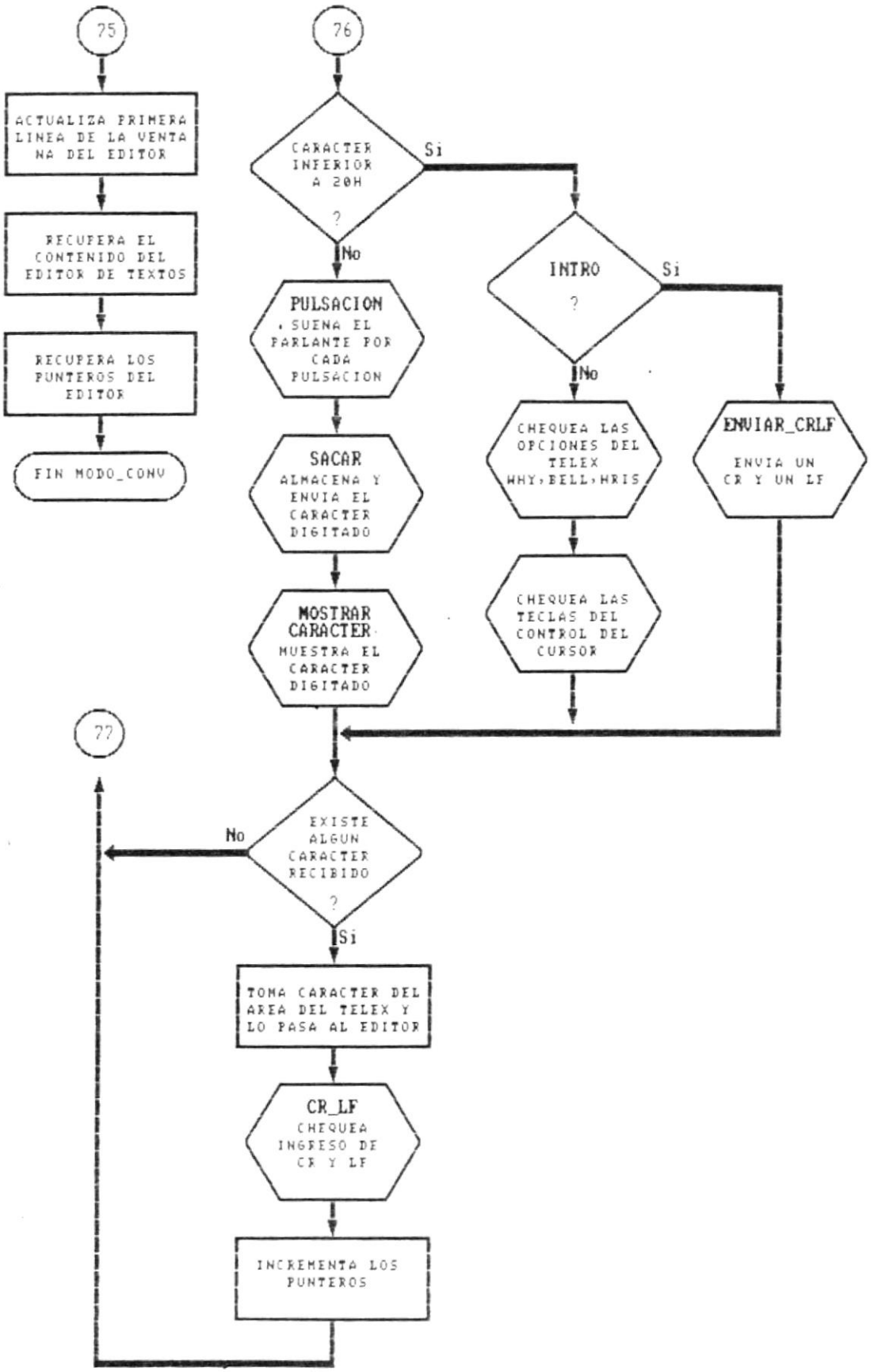

CONVERTIR
AL: CARACTER A CONVERTIR.
CX: COMPARACIONES
SI Y DI APUNTAN A LAS CADENAS ASCII Y BAUDOT
COMPARA CONTENIDO DE AL CON CADENA DE DI
EXISTE EL VALOR DE AL EN LA CADENA DE DI
No
Si
RETORNA CON ZF=1
CALCULA LA LA POSICION DEL CARACTER EN [SI]
TOMA EL CARACTER CONVERTIDO
RETORNA CON ZF=0
FIN CONVERTIR

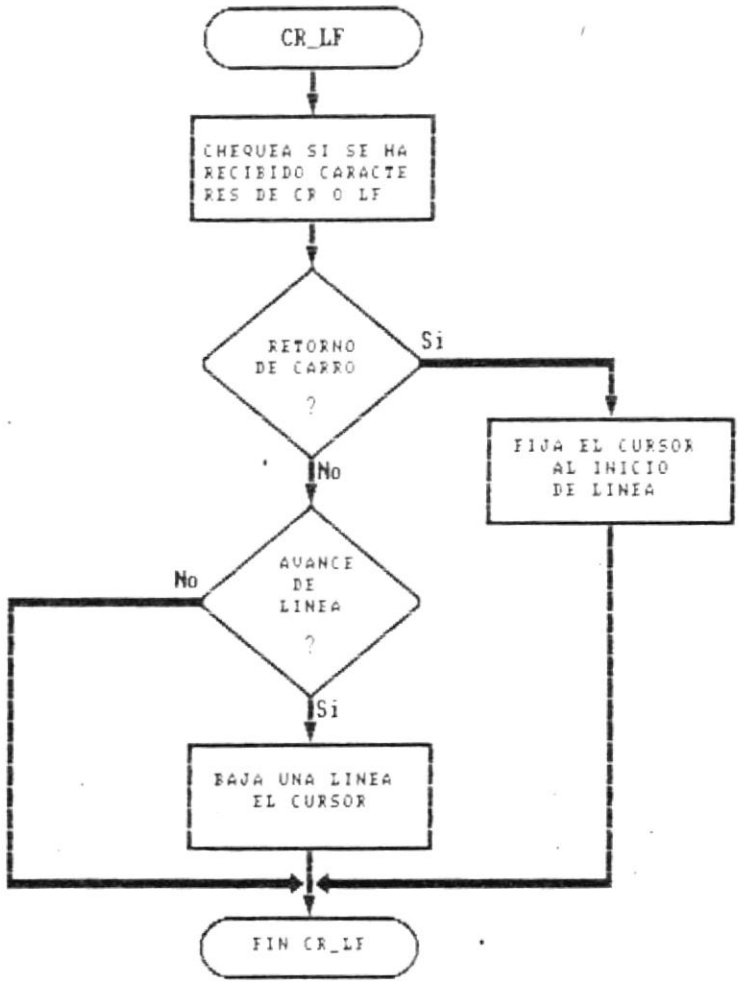

COLGAR
FIJA DX EN EL REGISTRO DE CONTROL DE MODEM
COM1: 3F8H + 4
COM2: 2F8H + 4
73

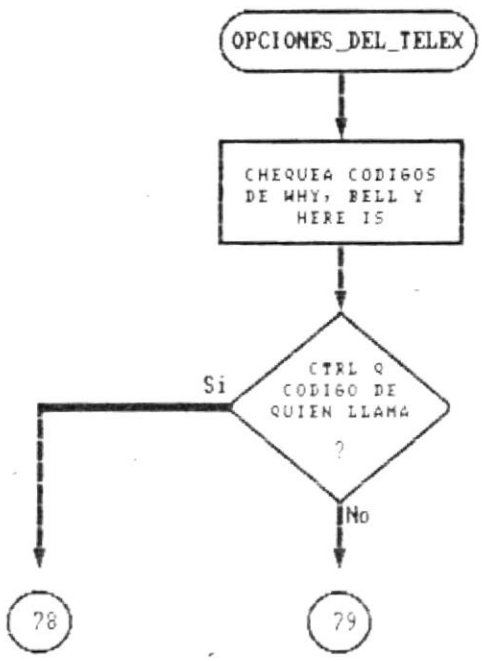

73
DTR Y RTS SON FORZADAS A NIVEL BAJO
CON ESTO IETEL DEBE INVERTIR LA LINEA DE TELEX
FIN COLGAR
YO_SOY
ENVIAR_CRLF
PRIMERO ENVIA UN CR Y UN LF
FIJA SI EN LA CADENA QUE POSEE EL INDICATIVO
TOMA CARACTER DEL INDICATIVO
SACAR
ENVIA EL CARACTER
MOSTRAR CARACTER
MUESTRA EL CARACTER ENVIADO
FIN DEL INDICATIVO
No
Si
FIN YO_SOY

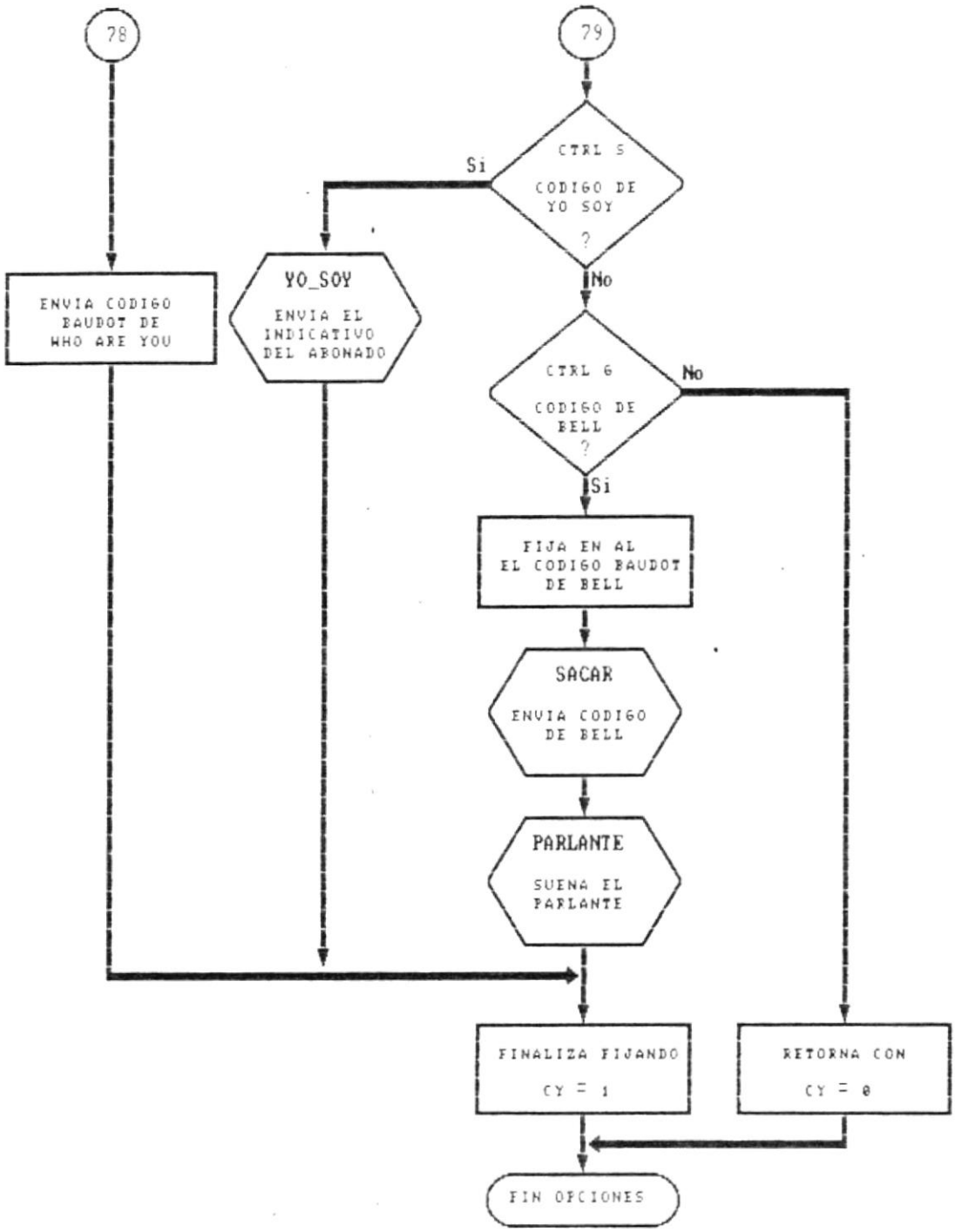

ENVIAR_CRLF
ENVIAR CARACTER
CODIGO BAUDOT DE RETORNO DE CARRO
MOSTRAR CARACTER
CODIGO ASCII DE RETORNO DE CARRO
ENVIAR CARACTER
CODIGO BAUDOT DE AVANCE DE LINEA
MOSTRAR CARACTER
CODIGO ASCII DE AVANCE DE LINEA
FIN ENVIAR_CRLF

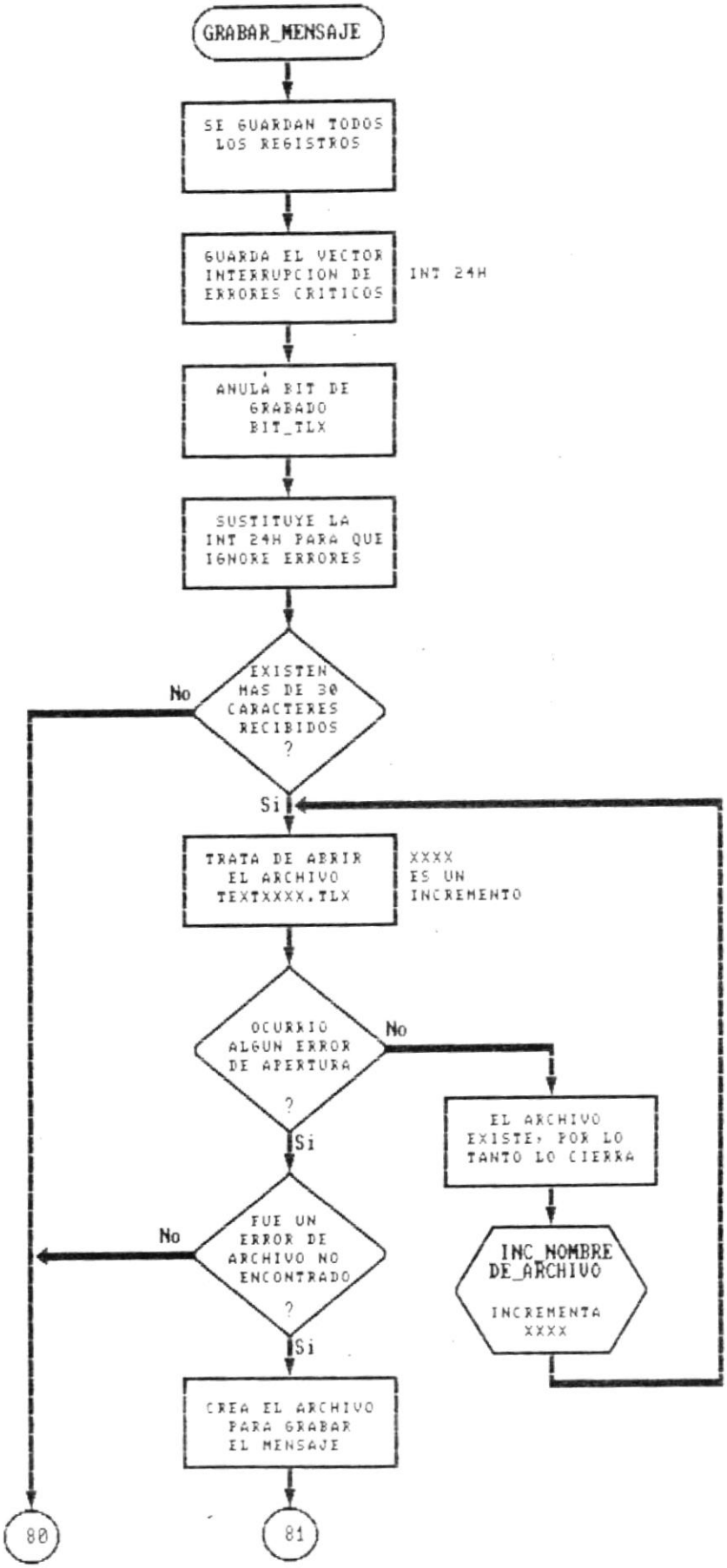

MODO_CONVERSACIONAL
GUARDA LOS PUNTEROS DEL EDITOR DE TEXTOS
GUARDA EL CONTENIDO DEL EDITOR
BORRAR_AREA TXT
BORRA EL ESPACIO DEL EDITOR

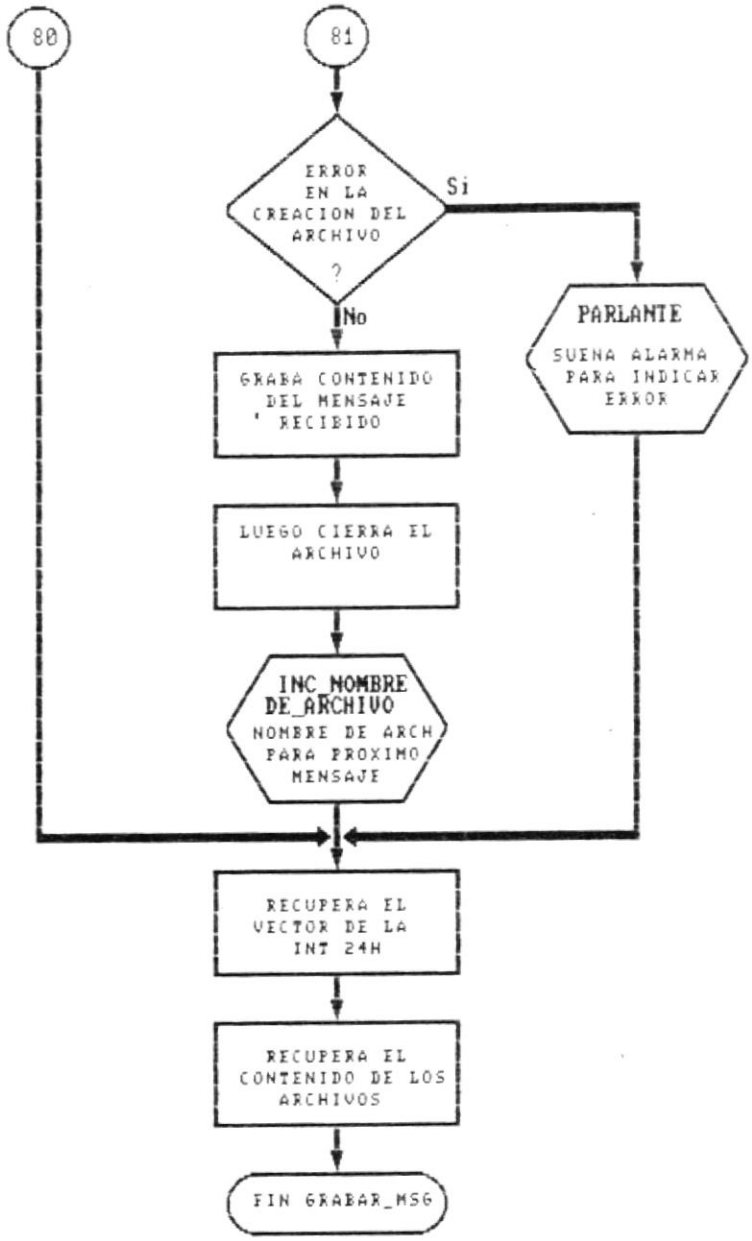

75
ACTUALIZA PRIMERA LINEA DE LA VENTANA DEL EDITOR
RECUPERA EL CONTENIDO DEL EDITOR DE TEXTOS
RECUPERA LOS PUNTEROS DEL EDITOR
FIN MODO_CONV
76
CARACTER INFERIOR A 20H ?
Si
No
INTRO ?
Si
No
PULSACION
, SUENA EL PARLANTE POR CADA PULSACION
SACAR
ALMACENA Y ENVIA EL CARACTER DIGITADO
MOSTRAR CARACTER
MUESTRA EL CARACTER DIGITADO
CHEQUEA LAS OPCIONES DEL TELEX WHY, BELL, HRIS
CHEQUEA LAS TECLAS DEL CONTROL DEL CURSOR
ENVIAR_CRLF
ENVIA UN CR Y UN LF
77
EXISTE ALGUN CARACTER RECIBIDO ?
No
Si
TOMA CARACTER DEL AREA DEL TELEX Y LO PASA AL EDITOR
CR_LF
CHEQUEA INGRESO DE CR Y LF
INCREMENTA LOS PUNTEROS

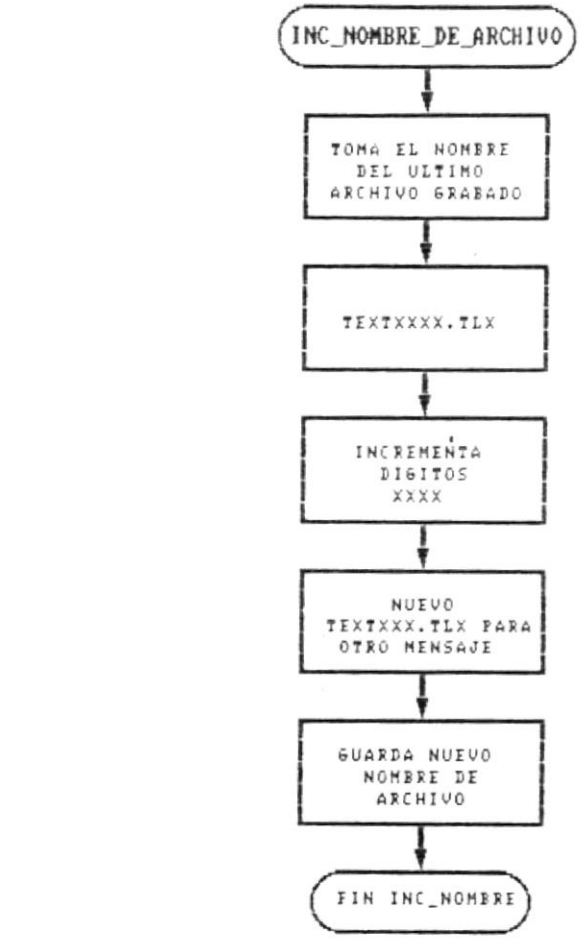

CR_LF
CHEQUEA SI SE HA RECIBIDO CARACTE RES DE CR O LF
RETORNO DE CARRO ?
Si
FIJA EL CURSOR AL INICIO DE LINEA
No
AVANCE DE LINEA ?
No
Si
BAJA UNA LINEA EL CURSOR
FIN CR_LF

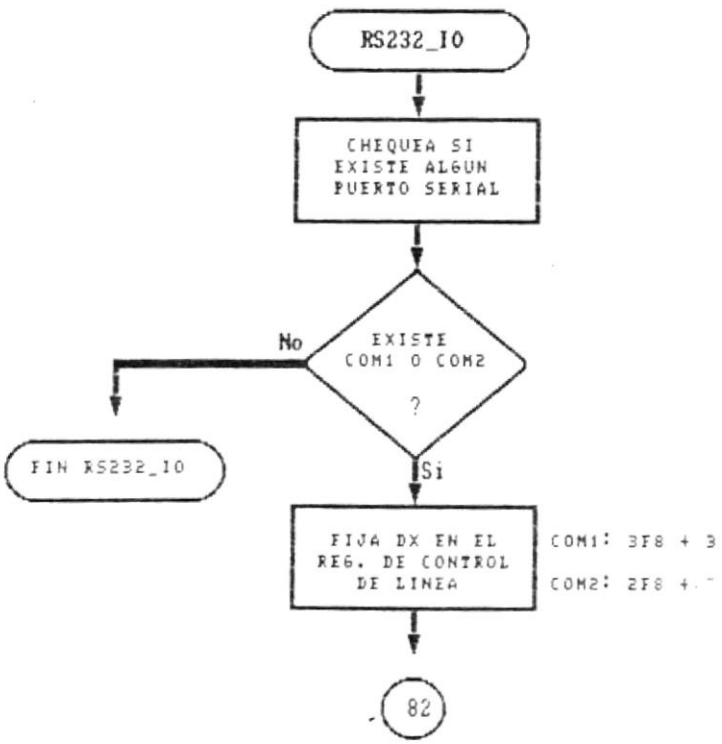

OPCIONES_DEL_TELEX
CHEQUEA CODIGOS DE WHY, BELL Y HERE IS
CTRL Q CODIGO DE QUIEN LLAMA ?
Si
No
78
79

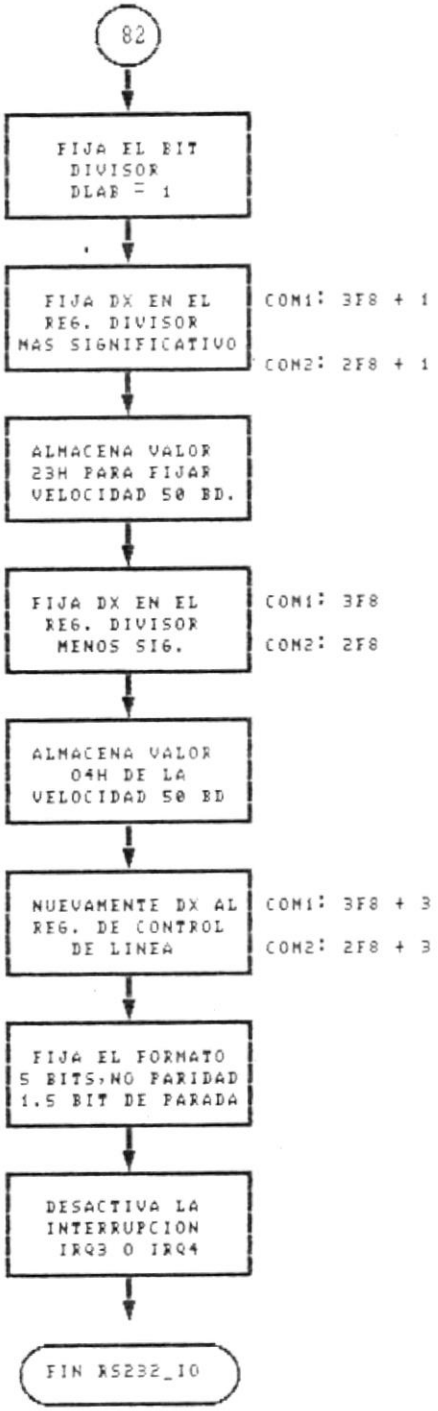

78
79
CTRL S
CODIGO DE
YO SOY
?
Si
No
YO_SOY
ENVIA EL
INDICATIVO
DEL ABONADO
ENVIA CODIGO
BAUDOT DE
WHO ARE YOU
CTRL G
CODIGO DE
BELL
?
No
Si
FIJA EN AL
EL CODIGO BAUDOT
DE BELL
SACAR
ENVIA CODIGO
DE BELL
PARLANTE
SUENA EL
PARLANTE
FINALIZA FIJANDO
CY = 1
RETORNA CON
CY = 0
FIN OPCIONES

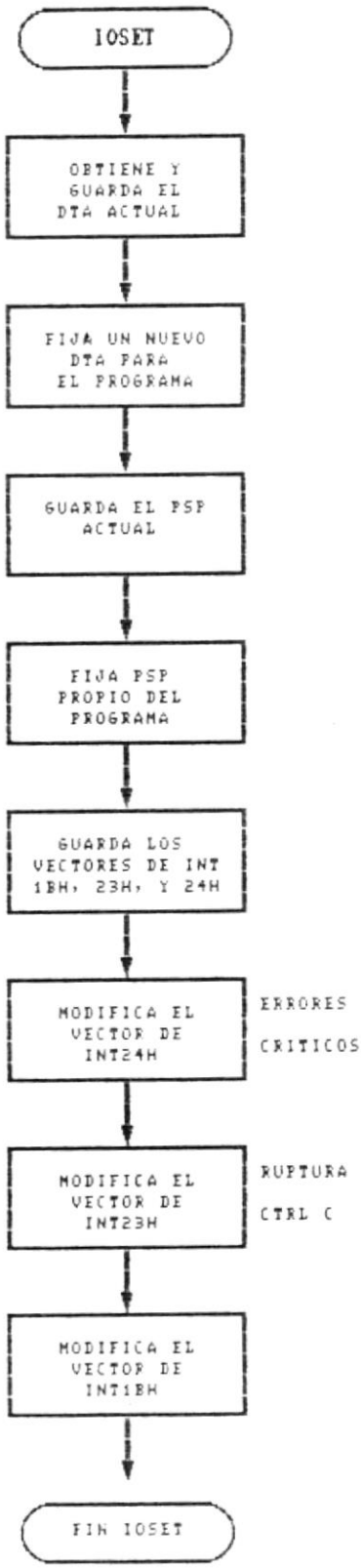

GRABAR_MENSAJE
SE GUARDAN TODOS LOS REGISTROS
GUARDA EL VECTOR INTERRUPCION DE ERRORES CRITICOS
INT 24H
ANULA BIT DE GRABADO BIT_TLX
SUSTITUYE LA INT 24H PARA QUE IGNORE ERRORES
EXISTEN MAS DE 30 CARACTERES RECIBIDOS ?
No
Si
TRATA DE ABRIR EL ARCHIVO TEXTXXXX.TLX
XXXX ES UN INCREMENTO
OCURRIO ALGUN ERROR DE APERTURA ?
No
Si
EL ARCHIVO EXISTE, POR LO TANTO LO CIERRA
INC_NOMBRE DE_ARCHIVO
INCREMENTA XXXX
FUE UN ERROR DE ARCHIVO NO ENCONTRADO ?
No
Si
CREA EL ARCHIVO PARA GRABAR EL MENSAJE
80
81


80
81
ERROR EN LA CREACION DEL ARCHIVO ?
Si
No
PARLANTE
SUENA ALARMA PARA INDICAR ERROR
GRABA CONTENIDO DEL MENSAJE ' RECIBIDO
LUEGO CIERRA EL ARCHIVO
INC_NOMBRE DE_ARCHIVO
NOMBRE DE ARCH PARA PROXIMO MENSAJE
RECUPERA EL VECTOR DE LA INT 24H
RECUPERA EL CONTENIDO DE LOS ARCHIVOS
FIN GRABAR_MSG


INC_NOMBRE_DE_ARCHIVO
TOMA EL NOMBRE DEL ULTIMO ARCHIVO GRABADO
TEXTXXXX.TLX
INCREMENTA DIGITOS XXXX
NUEVO TEXTXXX.TLX PARA OTRO MENSAJE
GUARDA NUEVO NOMBRE DE ARCHIVO
FIN INC_NOMBRE


RS232_IO
CHEQUEA SI EXISTE ALGUN PUERTO SERIAL
EXISTE COM1 O COM2 ?
No
Si
FIN RS232_IO
FIJA DX EN EL REG. DE CONTROL DE LINEA
COM1: 3F8 + 3
COM2: 2F8 +


82
FIJA EL BIT DIVISOR DLAB = 1
FIJA DX EN EL REG. DIVISOR MAS SIGNIFICATIVO
COM1: 3F8 + 1
COM2: 2F8 + 1
ALMACENA VALOR 23H PARA FIJAR VELOCIDAD 50 BD.
FIJA DX EN EL REG. DIVISOR MENOS SIG.
COM1: 3F8
COM2: 2F8
ALMACENA VALOR 04H DE LA VELOCIDAD 50 BD
NUEVAMENTE DX AL REG. DE CONTROL DE LINEA
COM1: 3F8 + 3
COM2: 2F8 + 3
FIJA EL FORMATO 5 BITS,NO PARIDAD 1.5 BIT DE PARADA
DESACTIVA LA INTERRUPCION IRQ3 O IRQ4
FIN RS232_IO


IOSET
OBTIENE Y GUARDA EL DTA ACTUAL
FIJA UN NUEVO DTA PARA EL PROGRAMA
GUARDA EL PSP ACTUAL
FIJA PSP PROPIO DEL PROGRAMA
GUARDA LOS VECTORES DE INT 1BH, 23H, Y 24H
MODIFICA EL VECTOR DE INT24H
ERRORES CRITICOS
MODIFICA EL VECTOR DE INT23H
RUPTURA CTRL C
MODIFICA EL VECTOR DE INT1BH
FIN IOSET


IORESET
RECUPERA EL PSP DEL PROGRAMA INTERCEPTADO
RECUPERA EL AREA DE TRANFERENCIA DEL DISCO DTA
RECUPERA EL VECTOR DE LA INTERRUPCION 24H
RECUPERA EL VECTOR DE LA INTERRUPCION 23H
RECUPERA EL VECTOR DE LA INTERRUPCION 1BH
FIN IO_RESET

```
;
;  DATOS.LIB  contiene todas  las  constantes  y  variables
;  usadas durante la ejecución del programa residente.
;

version      db      13,10,'Telex en una PC/XT/AT 1.0',13,10
             db      'ESPOL-GYE 1991. Por Freddy Pinto Carpio'
             db      13,10
             db      'Secuencia de entrada: ALT-SHIFT izq.'
             db      13,10,'$'
alt_shift    equ     00001010b          ;Secuencia de entrada:
                                        ;ALT_SHIFT_IZQ
reloj        db      ?,?,'58,?,?,58,?,?,32,?,109
contador     db      1
espera       db      0

tiempo       struc
             hrs     db      ?
             min     db      ?
             sec     db      ?
tiempo       ends

tiempo_actual        tiempo  <>

kb_dato      equ     60h
duracion     dw      7FFFh              ;Duración del sonido del
                                        ;parlante
enter        equ     13                 ;Códigos de teclas INTRO
ESC          equ     27                 ;y ESC
segmento_dos dw      ?                  ;Segmento usado por el
                                        ;DOS.
indos        dw      ?                  ;Bandera interna del DOS
errflag      dw      ?                  ;Bandera Error Crítico
indicadores  db      0                  ;Bit's definidos como:
                                        ; 0 - Editor activo
                                        ; 1 - BIOS activo
                                        ; 2 - Disco activo
                                        ; 5 - DIR activo
                                        ; 6 - Ed. en preparación
                                        ; 7 - ALT SHIFT IZQ
                                        ;     presionadas
direc_6845   dw      ?                  ;Dirección base del CRT
valores_CGA  db      2ch,28h,2Dh,29h    ;Valores para habilitar
             db      2Ah,2Eh,1Eh        ;la salida del adaptador
                                        ;de color (CGA).
registro_ss  dw      ?                  ;Variables para almacenar
registro_sp  dw      ?                  ;contenido de  registros
                                        ;SS y SP
antiguo_psp  dw      ?                  ;Segmento del PSP del prg
                                        ;interrumpido
valor_ascii  db      '0000'             ;Usado para almacenar
                                        ;valores ASCII

dir_Grb_Imp_Env      dw      ?          ;Dirección desde donde se
```

```
                                        ;Grabará, Imprimirá o se
                                        ;Enviará un texto.
num_bytes    dw      ?                  ;Número de bytes a Grabar,
                                        ;Imprimir o Enviar

cursor_texto  dw inicio_area_de_textos ;Puntero cursor EDITOR

fin_del_texto dw inicio_area_de_textos ;Posición final del
                                        ;texto

inicio_pagina  dw inicio_area_de_textos ;Texto/ventana video

direc_video     dw   160*5              ;Posición inicial de la
                                        ;ventana de video
adaptador       db   ?                  ;Tipo de adaptador de
                                        ;video instalado
                                        ;0=MDA, 1=CGA, 2=EGA

sgmnto_de_video dw   0B000h             ;Segmento de Video
                                        ;B000h=Monocromático,
                                        ;B800h=Color


;
;                    Atributos
;

normal              db      ?       ;Caracteres normales
invertido           db      ?       ;Caracteres invertidos
attr_de_texto       db      ?       ;Color y fondo de los
                                    ;caracteres de texto
attr_de_dir         db      ?       ;Ventana del directorio
attr_de_ver         db      ?       ;Contenido de un archivo
                                    ;desde el directorio
at_brd_txt          db      ?       ;Borde de la ventana de
                                    ;textos
at_brd_dir          db      ?       ;Borde de la ventana del
                                    ;directorio
at_brd_menu         db      ?       ;Ventana del menú.
at_info_1           db      ?
at_info_2           db      ?
menu_attr_1         db      ?
menu_attr_2         db      ?

attr_del_reloj      db      ?       ;Atributos caracteres del
                                    ;reloj

lincol_attr         db      ?       ;LIN: COL:
pagina_de_video     db      ?       ;Página de video actual
modo_del_cursor     dw      ?       ;Forma del cursor
pscion_del_cursor   dw      ?       ;Posición del cursor
modo_de_video       db      ?       ;Modo de video actual
modo_erroneo        db      0       ;Error del modo de video
atributo            db      ?
```

```
;
;  Atributos para los caracteres a mostrar en las diferentes
;  ventanas proporcionadas por TELEX.
;

; Atributos para color

attr_para_color     db      0   ,0B8h ,70h ,1Eh  ,17h
                    db      70h ,3Fh  ,1Eh ,7Ah  ,11
                    db      3Fh ,01   ,1Eh ,01   ,30h
                    db      31h

; Atributos para monocromático.

attr_para_monoc     db      0   ,0B0h ,07  ,70h  ,07
                    db      07  ,0Fh  ,70h ,70h  ,70h
                    db      70h ,0    ,01  ,08   ,70h
                    db      70h

sendero             db      36 dup (0) ;Sendero del directorio
menu                dw      offset menu0

;
;              Mensajes enviados por el EDITOR
;

menu0       db      '$F1$-Dir $F2$-Leer $F3$-Grabar '
            db      '$F4$-Impr. $F5$-Borrar $F6$-Dial '
            db      '$F7$-Send $F8$-Stop Send $F10$-más',0
menu1       db      '$F9$-Cortar $^V$-Conversar '
            db      '$^Q$-Quién llama? $^S$-Yo soy: '
            db      '$^G$-Alarma $Sh F1$-Cnfg $ESC:$Salir',0
menu2       db      '┤Texto:$ ',52 dup (32),
            db      '$Línea:    Col:   ├'
menu5       db      'Grabar archivo:$ ',35 dup (32),0
menu6       db      'Leer archivo:$ ',0
menu7       db      'Error grabando archivo...',0
menu8       db      'Espacio insuficiente en el disco...',0
menu9       db      'Archivo no encontrado...',0
menu10      db      'Imprimiendo...
            db      '$[P]$ pausa',0
menu11      db      'Error imprimiendo texto....       ',0
menu12      db      'Presione $INTRO$ para continuar '
            db      'o $ESC$ para salir',0
menu13      db      '¿Borrar Texto? $(S/N) ',0
menu14      db      'No existe texto alguno para Grabar, '
            db      'Imprimir, Borrar o Enviar...',0
menu15      db      '$Prepare impresora...
            db      ;Presione $INTRO$ para imprimir '
```

```
;
;   Códigos de teclas para el control del cursor
;

teclas_editor        label   byte

                     db      4Dh,4Bh,48h
                     db      50h,49h,51h
                     db      84h,76h,1Ch
                     db      47h,4Fh,44h
                     db      54h,40h,41h
                     db      43h,0Eh,0Fh
                     db      52h,53h,5Dh
                     db      71h,3Bh,3Ch
                     db      3Dh,3Eh,3Fh

subroutinas_del_editor       label   word

            dw       avz_cursor      ,ret_cursor   ,subir_cursor
            dw       bajar_cursor    ,subir_pg     ,bajar_pg
            dw       inicio_pg       ,fin_pg       ,intro
            dw       inicio_de_linea ,fin_de_linea  ,mas
            dw       configurar      ,marcar       ,enviar
            dw       colgar          ,bckspc       ,tab
            dw       ins_caracter ,borrar_caracter ,ins_linea
            dw       borrar_linea ,directorio   ,leer_archivo
            dw       grabar          ,imprimir     ,borrar


;
;   Códigos ASCII de vocales con tilde y eñe's
;

vocales_tildadas  label     byte

                  db        0A0h,082h,0A1h          ;á é í
                  db        0A2h,0A3h,0A4h          ;ó ú ñ
                  db        0A5h                    ;Ñ

;
;   Códigos de teclas asignados para formar vocales con
;   tildes y eñe's.
;

teclas_de_vocales label     byte

                  db  1,5             ;CTRL-A[a], CTRL-E[e],
                  db  9,0Fh           ;CTRL-I[i], CTRL-O[o]
                  db  15h,0Eh         ;CTRL-U[u], CTRL-N[n]
                  db  0Dh             ;CTRL-M[m]
```

```
;
;   Tabla del Código BAUDOT (n/a = No asignado)
;

lfsh_out    dw      0               ;0=Mayúsculas Código
                                    ;Baudot.
                                    ;26 Minúsculas
lfsh_in     dw      0
                                    ; Abecedario   Numéricos

baudot      label       byte

        db  03, 25, 14, 09, 01      ; A B C D E   - ? : WHY 3
        db  13, 26, 20              ; F G H I J   n/a n/a n/a
        db  06, 11                  ;             8 BELL
        db  15, 18, 28, 12, 24      ; K L M N O   ( ) . , 9
        db  22, 23, 10, 05, 16      ; P Q R S T   0 1 4 , 5
        db  07, 30, 19, 29, 21      ; U V W X Y   7 = 2 / 6
        db  17                      ; Z             +
        db  04                      ; Espacio en blanco inv.
        db  00                      ; Espacio en blanco.
        db  08                      ; Retorno de carro (CR)
        db  02                      ; Avance de línea (LF)
LSH     db  31                      ; Abecedario
FSH     db  27                      ; Numéricos

;
;   Tabla del Código ASCII (n/a = No asignado)
;

                                    ; Abecedario
ascii       label   byte

            db      65, 66, 67, 68, 69      ; A B C D E
            db      70, 71, 72, 73, 74      ; F G H I J
            db      75, 76, 77, 78, 79      ; K L M N O
            db      80, 81, 82, 83, 84      ; P Q R S T
            db      85, 86, 87, 88, 89      ; U V W X Y
            db      90                      ; Z

                                    ; Numéricos

            db      45, 63, 58, 17, 51      ; - ? : WHY 3
            db      00, 00, 00, 56, 07      ; 00=n/a 8 BELL
            db      40, 41, 46, 44, 57      ; ( ) . , 9
            db      48, 49, 52, 44, 53      ; 0 1 4 , 5
            db      55, 61, 50, 47, 54      ; 7 = 2 / 6
            db      43                      ; +
            db      32                      ; Nulo
            db      32                      ; Espacio blanco
            db      13                      ; Retorno carro
            db      10                      ; Avance linea
```

```
;
;                Variables usadas por el DIRECTORIO
;

disco                   db      ?
disco_actual            db      ?       ;Unidad de disco actual.
cur_file                dw      ?
fin_offset              dw      ?
paginas                 dw      ?
senal_verific           db      1       ;Señal de verificación.
tabla_de_variab         dw      18*160+485, 1   ;Variables.
                        dw      485, nombres_de_archivos,0
                        db      0, 70h, 0
fin_de_pagina           dw      18*160+485      ;Final de página
numero_de_archvs        dw      1       ;Número de archivos.
linea_actual            dw      485
cur_offset              dw      nombres_de_archivos
orden_offset            dw      0
indicador_orden         db      0       ;Bandera de ordenamiento.
                                        ;0 = Ordenar por nombres.
                                        ;1 = Ordenar por fecha.

attr_barra              db      70h     ;Atributo de la barra
puntero_de_linea        db      0       ;del directorio.
global                  db      '*.*',0 ;Todos los archivos.
indicador_video         db      ?
fin_de_archivo          dw      ?       ;Fin de archivo.
puntero_del_archivo     dw      ?       ;Manipulador de archivo.
ultima_pagina           dw      ?       ;Ultima página.


tabla_dir               db      1   ,1ch ,3bh ,3ch
                        db      3dh ,3eh ,3fh ,40h
                        db      41h ,43h ,44h ,47h
                        db      48h ,49h ,4fh ,50h
                        db      51h ,76h ,84h


subroutinas_dir         label word

            dw      salirx  ,ver_archivo ,ver_archivo
            dw      editar_archivo  ,renombrar ,mover_archivo
            dw      borrar_archivo  ,confirmar
            dw      cambiar_de_directorio ,ordenar_por_nombre
            dw      ordenar_por_fecha ,home_bar  ,subir_barra
            dw      pg_upx  ,fin_bar   ,bajar_barra   ,pg_dnx
            dw      bottom_bar,top_bar

view_table  dw      home_file ,up_line ,up_pg     ,fin_file
            dw      dn_line   ,dn_pg   ,fin_file ,home_file

view_menu   db      '$F1$-Salir $F2$-Editar $F3$-Renombrar'
            db      ' $F4$-Mover $F5$-Borrar $F7$-Cambiar de'
            db      ' DIRECTORIO $ESC$',0
```

```
;
;          Mensajes enviados por el DIRECTORIO
;

dir_erroneo     db   'Directorio erróneo...',0
dir_vacio       db   'No se encontraron archivos...     ',0
demasiados      db   'Existen  demasiados  archivos...',0
leyendo         db   'Leyendo y Ordenando directorio...',0
directorio_de   db   'Directorio de ',0
cambio_de_dir   db   'Cambio de directorio:',0
archivos        db   '    Archivo(s)',0
sera_borrado    db   ' será borrado.',0,'Está seguro? S/N',0
sera_editado    db   ' será editado.',0,'Quiere hacerlo? S/N'
                db   0
nuevo_nombre    db   'Nuevo nombre para ',0
nuevo_dir       db   'Nuevo directorio para ',0
on              db   'SI',0
off             db   'NO',0
codigos_corregir    db      8,32,8,0
cuatro_Kbytes       equ     4096

;
;     Contenido de la ventana del Menú del Directorio.
;

menu_dir      label byte

    db    '┤ Directorio TELEX ├'
    db    ' F.I.E - ESPOL 1989 ',' $Freddy Pinto C.$   '
    db    20 dup (196),' $F1$   Ver  ( o ',17,196,217,') '
    db    ' $F2$   Editar        ',' $F3$   Renombrar     '
    db    ' $F4$   Mover         ',' $F5$   Borrar        '
    db    ' $F6$   Confirmar  $SI$ ',' $F7$   Cambiar de DIR'
    db    ' $F9$   Ord. por Nom. ',' $F10$  Ord. por Fecha'
    db    ' $ESC$  para salir    '
    db    20 dup (205),' $Use:$ ',24,32,25,' PgUp PgDn '
    db    '^PgUp ^PgDn Home End',20 dup (32)

;
;          Contenido de la ventana de Configuración.
;

menu_cnfg   label   byte

            db      '┤Configuración  TELEX - PC'XT/AT├'
port        db      ' $Puerto Serial:$ '
            db      'COM1  COM2  -NO-  '
baudios     db      ' $Baudios:$  '
            db      '50   75  100          '
            db      ' $Dir  TELEX:$ '
```

```
dir_telex    db      'c:\telex',12 dup (0)
             db      ' $Indicativo:$ '
indicativo   db      '43509 ESPOLG ED',5 dup (0)

bauds        dw      2304            ;2304:   50 baudios
                                     ;1646:   70 baudios
                                     ;1152:  100 baudios

bits         db      00000100b       ;5N1.5:-5 bits de datos
                                     ;      -No paridad
                                     ;      -1.5 bit de parada
cursor_cnfg          dw      ?
puntero_cnfg         dw      ?
campo                db    ' 0,10,10
puntero_de_telex     dw      area_de_telex
puntero_anterior     dw      area_de_telex

puerto_serial        dw      0       ;Puerto de datos.
                                     ;COM1: 3F8h
                                     ;COM2: 2F8h
bits_tlx             db      0       ;Bits definidos como:
                                     ; 2 - Grabar mensaje
                                     ;     enviado/recibido.
segundo              db      0
time_tlx             dw      18*60*3 ;3 minutos
en_linea             db      0       ;Indicador del estado de
                                     ;la línea.
indicador_tlx        db      ' '
CTS_DSR              db      ?
archivo_de_telex     db      'TEXT'
                     db      4 dup (48)
                     db      '.TLX',0

;[d:][\directorio][\archivo_de_telex]

refile               db      30 dup (0)
```

```
;
;       TELEX.COM para la IBM Computadora Personal 1991.
;                   Por Freddy Pinto Carpio.
;

BIOS_SEG    SEGMENT AT      40H

rs232_base  dw 4 dup (?)            ;Dirección del estado del
                                    ;puerto serial RS-232C
    org     17h                     ;Estado de ALT-CTRL-SHIFT
kb_flag     db      ?               ;del teclado
    org     4ah                     ;Número de columnas en
crt_cols    dw      ?               ;video
    org     87h                     ;Estado de la tarjeta EGA
estado_EGA  db      ?               ;0= EGA en Color
                                    ;1= EGA en Monocromático
BIOS_SEG    ENDS

CODE    SEGMENT     PARA PUBLIC 'CODE'

    assume cs:code
    org     100h                    ;Todo programa .COM debe
                                    ;comenzar en el offset
                                    ;100h
BEGIN:
    jmp     inicio                  ;Preparación
    include datos.lib               ;Constantes y variables
                                    ;usadas por el programa

;
;                       TECLADO
;
;               Subroutina de control del teclado
;
;  La ejecución comienza aquí por medio de la Interrupción
;  9h cada vez que una tecla es presionada. Si esta
;  subroutina detecta que las teclas ALT-SHIFT-IZQ están
;  siendo presionadas, fija el bit 7 de la variable
;  INDICADORES de modo que las subroutinas BACKPROC o TIMER
;  traten de activar la ventana del Editor.
;

TECLADO     PROC    NEAR
    push    ax                      ;AX y DS serán modificados
    push    ds                      ;por lo tanto se reservan
    sti                             ;Habilita interrupciones
    push    cs                      ;DS al seg. del programa.
    pop     ds
    test    indicadores,01000001b   ;¿ Editor ya activado?
    jnz     evitar_reset            ;Si, evitar ALT-CTRL-DEL
    pushf                           ;No, entonces llama a la
```

```
    call    teclado_int             ;subroutina original
    push    cs
    pop     ds
    mov     ah,2                    ;Toma el estado de SHIFT
    int     16h                     ;del teclado
    and     al,0Fh                  ;Anula los 4 bits MSB
    cmp     al,ALT_SHIFT            ;¿ Teclas ALT-SHIFT-IZQ
                                    ;presionadas ?
    jne     salir_kb                ;No, salir
    or      indicadores,10000000b   ;Si, entonces fija BIT 7
                                    ;para que las subroutinas
                                    ;TIMER o BACKPROC lo
                                    ;reconozcan.
    jmp     short salir_kb          ;Finalmente sale

;Se evita la secuencia ALT-CTRL-DEL únicamente cuando la
;subroutina del TELEX esté activada.


evitar_reset:

    assume  ds:bios_seg
    mov     ax,bios_seg
    mov     ds,ax                   ;DS al segmento del BIOS
    mov     ah,kb_flag
    and     ah,0Ch
    cmp     ah,0Ch                  ;¿ Secuencia ALT-CTRL ?
    jne     kb
    in      al,kb_dato
    cmp     al,53h                  ;¿ Tecla DEL ?
    jne     kb
    and     kb_flag,0f3h            ;Si, entonces anula la
kb:                                 ;secuencia ALT-CTRL
    pushf                           ;Simula una interrupción
    call    teclado_int             ;y llama a la subroutina
                                    ;original.
salir_kb:
    pop     ds                      ;Recupera los registros
    pop     ax                      ;modificados
    assume  ds:nothing
    iret
TECLADO     ENDP


;
;  Nueva subroutina de PRINT SCREEN: Evita la impresión de
;  la pantalla cuando el Editor esté activado.
;


INT05       PROC    FAR
    push    ds
    push    cs
    pop     ds                      ;DS=CS
```

```
    test    indicadores,00000001b   ;¿ Editor activado ?
    jnz     fin05                   ;Si, ignorar el pedido de
                                    ;impresión
    pushf                           ;Caso contrario, simula
                                    ;una INT 05
    call    prtsc_int               ;Llama a la subroutina
                                    ;del BIOS.
fin05:                              ;que imprime la pantalla.
    pop     ds                      ;Recupara el registro DS.
    iret
INT05       ENDP

;
;   TIMER
;   Subroutina del manejo de la interrupción 8h. Reloj
;   principal del sistema.
;

TIMER       PROC    NEAR
    pushf                           ;Simula una interrupción
    call    timer_int               ;8h
    cmp     cs:espera,0
    je      ck_alt_shift
    dec     cs:espera
    and     cs:indicadores,00111111b
    iret
ck_alt_shift:
    cmp   cs:indicadores,10000000b  ;¿ALT-SHIFT Izq presentes
                                    ;cuando existen accesos
                                    ;al BIOS o al DISCO ?
    jne    salir_timer              ;Si, salir
    xor    cs:indicadores,11000000b ;No, anula el bit de
                                    ;ALT-SHIFT y fija el bit
                                    ;que indica que TELEX se
                                    ;activará
    push    es                      ;Chequea bandera interna
    push    di                      ;del DOS y la de Error
    mov     es,cs:segmento_dos      ;Crítico. Ambas deben
    assume  es:nothing              ;tener un valor cero
    mov     di,cs:indos
    cmp     byte ptr es:[di],0      ;¿ Bandera INDOS = 0 ?
    jne     poptime                 ;No, salir
    mov     di,cs:errflag
    cmp     byte ptr es:[di],0      ;¿ Error crítico = 0 ?
    jne     poptime                 ;No, salir
    mov     al,20h
    out     20h,al                  ;Caso contrario

    CALL    PRINCIPAL               ;Llama a la subroutina
                                    ;principal que activa al
    and     cs:espera,0             ;EDITOR para el telex
poptime:
    pop     di                      ;Al salir, recupera los
```

```
    pop     es                          ;registros ES y DI
salir_timer:
    and     cs:indicadores,00111111b
    iret                                ;Retorna
TIMER       ENDP


;
;   BACKPROC
;   Subroutina del manejo de la interrupción 28h.
;

BACKPROC    PROC    NEAR
    pushf                               ;Llama a la subroutina
    call    bp_int                      ;original INT 28h
    cmp     cs:indicadores,10000000b    ;¿ALT-SHIFT Izq presentes
                                        ;cuando existen accesos
                                        ;al BIOS o al DISCO ?
    jne     salir_bp                    ;Si, salir
    xor     cs:indicadores,11000000b    ;No, anula el bit de
                                        ;ALT-SHIFT y fija bit
                                        ;que indica que TELEX se
                                        ;activará
    push    es                          ;Guarda ES y DI
    push    di
    assume  es:nothing
    mov     es,cs:segmento_dos          ;Chequea la bandera de
    mov     di,cs:errflag               ;Error Crítico
    cmp     byte ptr es:[di],0          ;¿ Error crítico = 0 ?
    jne     popbp                       ;No, salir
                                        ;Si, entonces
    CALL    PRINCIPAL                   ;llama a la subroutina
                                        ;principal que activa el
    and     cs:espera,0                 ;Editor para el telex

popbp:

    pop     di                          ;Recupera los registros
    pop     es                          ;ES y DI
salir_bp:
    test    cs:bits_tlx,100b            ;¿Algún mensaje recibido?
    je      fin_bp                      ;No, salir
    call    grabar_mensaje              ;Caso contrario, graba el
fin_bp:                                 ;mensaje
    and     cs:indicadores,00111111b
    iret                                ;Retorna
BACKPROC    ENDP


;
;   VIDEO
;   Subroutina del manejo de la Interrupción 10h. Acceso a
;   los servicios de Video del BIOS.
;
```

```
VIDEO       PROC    NEAR
    pushf                               ;Simula una INT del BIOS.
    mov     cs:espera,18
    or      cs:indicadores,10b          ;Señal de que el BIOS
                                        ;está ocupado
    call    video_int                   ;Subroutina del BIOS
    and   cs:indicadores,11111101b      ;BIOS libre
    iret
VIDEO       ENDP


;
;                          BIOS_DISCO
;  Subroutina del manejó de la  Interrupción 13h. Servicios
;  disco estándar de la ROM-BIOS.
;


BIOS_DISCO  PROC    FAR
    pushf                               ;Simula INT del BIOS.
    mov     cs:espera,18
    or      cs:indicadores,100b         ;Señal de que se está
                                        ;ejecutando un acceso al
                                        ;disco
    call    bios_disco_int              ;Llama a la subroutina
    pushf                               ;del BIOS
    and   cs:indicadores,11111011b      ;BIOS libre
    popf
    sti
    ret     2                           ;Retorna sin destruir los
BIOS_DISCO  ENDP                        ;indicadores de estado


;
;                             INTIC
;  Subroutina del  manejo  de  la  Interrupción  1CH.
;  Interrupción generada por la INT 8h,18 veces por segundo.
;
;  Esta subroutina  es la parte  principal  de  TELEX. Ella
;  chequea  el  puerto  de  comunicación,  y  cada  vez que
;  encuentra un  caracter disponible, lo toma y lo almacena
;  en  el  espacio  de  memoria  reservado para mensajes de
;  telex recibidos. Esta  misma  subroutina, se  encarga de
;  mostrar el  reloj  en modo continuo únicamente cuando la
;  ventana del Editor esté activada.
;


INTIC       PROC    NEAR
    push    ax                          ;Se guardan todos los
    push    bx                          ;registros usados por
    push    cx                          ;esta subroutina
    push    dx
    push    si
    push    di
```

```
        push    ds
        push    es
        mov     ax,cs
        mov     es,ax                   ;DS y ES al segmento del
        mov     ds,ax                   ;programa.
        mov     dx,puerto_serial
        cmp     dx,0
        je      ck_EDITOR
        call    ck_entrada_serial
ck_EDITOR:
        test    indicadores,1b          ;El reloj se envía a la
        jz      fin_intic               ;pantalla solo cuando el
        test     indicadores,100000b    ;EDITOR esté activado,
        jnz     fin_intic               ;pero que no se esté con
        mov     es,sgmnto_de_video      ;las ventanas DIR y MENU
        mov     di,160*5+53*2
        mov     bl,indicador_tlx        ;Estado de la línea del
                                        ;Telex.
        call    escribir_byte           ;Indica estado de CTS
                                        ;(Clear to send)
        call    actualizar_reloj        ;Luego, muestra el reloj
fin_intic:                              ;del EDITOR
        pop     es                      ;Recupera registros
        pop     ds                      ;usados
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        iret

INTIC   ENDP
```

```
;
;                              IO_ERROR
;  Subroutina de la Interrupción 24h (Solo para versiones
;  3.X) Manipuladora de errores críticos detectados por el
;  DOS.
;
;  Evita el mensaje 'Cancelar, Reintentar o Ignorar'.
;
;  Le indica al DOS que ignore todos los errores fatales
;  de disco, cuando éste programa trate de realizar tareas
;  de lectura o escritura al disco.
;
```

```
IO_ERROR    PROC    NEAR
        sti
        xor     al,al                   ;Indica al DOS que ignore
io_salida:                              ;el error
        iret
IO_ERROR    ENDP
```

```
;
;                                PRINCIPAL
;  Es la subroutina llamada  para  formar  y  manipular la
;  ventana de textos. A partir de esta subroutina se prepara
;  todo el panorama usado por TELEX.
;

PRINCIPAL   PROC    NEAR
    cli                                 ;Desactiva interrupciones
    mov     cs:registro_ss,ss           ;para guardar registros
    mov     cs:registro_sp,sp           ;de Pila
    push    cs                          ;Conmuta a una pila
    pop     ss                          ;interna de TELEX
    lea     sp,nueva_pila
    sti                                 ;Activa interrupciones
    push    ax                          ;y guarda los otros
    push    bx                          ;registros.
    push    cx                          ;ES y DI no se almacenan
    push    dx                          ;TIMER o BACKPROC ya  lo
    push    si                          ;hicieron
    push    ds
    push    bp

    push    cs                          ;Fija DS al segmento del
    pop     ds                          ;programa
    assume  ds:code
    call    ioset                       ;Cambia interrupciones
                                        ;1Bh, 23h y 24h
                                        ;Modifica DTA y PSP
    call    rs232_io                    ;Configura puerto serial
                                        ;a 50 bauds, 5 bits, no
                                        ;paridad, 1.5 bits de
                                        ;parada
    call    video_param                 ;Chequea los parámetros
                                        ;de Video
    or      indicadores,1               ;Señal que indica que la
                                        ;ventana del EDITOR está
                                        ;activada
    call    EDITOR                      ;Llama a la subroutina
                                        ;que ejecuta funciones de
                                        ;procesador de palabras,
                                        ;Directorio y Telex
    and     indicadores,11111110b       ;Al salir,
                                        ;limpia la bandera del
    call    recuperar_video_param       ;EDITOR y deja las cosas
                                        ;tal como estaban al
                                        ;inicio del programa.
    call    ioreset                     ;Recupera interrupciones
                                        ;1Bh, 23h 24h,
                                        ;DTA y PSP

; Recupera los registros y la pila antes de salir.

    pop     bp
```

```
        pop     ds
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        cli                             ;Desactiva interrupciones
        mov     ss,cs:registro_ss       ;Conmuta a la pila
        mov     sp,cs:registro_sp       ;original
        sti                             ;Activa interrupciones
        ret
PRINCIPAL   ENDP

;
;                          VIDEO_PARAM
;   Fija los valores de los diferentes parámetros requeridos
;   durante la ejecución del programa.
;

VIDEO_PARAM PROC    NEAR

; Guarda los parámetros de video que deberán ser usados ahora
; o después.

        mov     ah,15                   ;Coge el modo y página de
        int     10h                     ;video activos
        mov     modo_de_video,al        ;Guarda el modo de video
        mov     pagina_de_video,bh      ;y página visualización
        mov     ah,3                    ;Función del BIOS para
                                        ;leer la localización del
                                        ;cursor
        int     10h                     ;Servicio de video
        mov     modo_del_cursor,cx      ;Guarda la forma y la
        mov     pscion_del_cursor,dx    ;posición del cursor
        cmp     adaptador,1             ;Si existe una CGA, se
        jne     guardar_scr             ;desactiva la salida de
        call    desactivar_CGA          ;video
guardar_scr:
        call    guardar_video           ;Guarda el contenido de
                                        ;la pantalla
        cmp     modo_de_video,2         ;¿Modo 2? 80 x 25 Monoc.
        je      abrir_ventana           ;Si, continúe
        cmp     modo_de_video,3         ;¿Modo 3? 80 x 25 Color
        je      abrir_ventana           ;Si, continúe
        cmp     modo_de_video,7         ;¿Modo 7? 80 x 25 Monoc.
        je      abrir_ventana           ;Si, abrir la ventana del
                                        ;EDITOR

        or      modo_erroneo,1          ;No, señal que indica que
                                        ;el mode de video será
                                        ;cambiado.
        xor     ah,ah                   ;Ajusta al modo de texto
                                        ;80 x 25
```

```
    mov     al,7                    ;Modo 7 para monoc.
    test    cs:adaptador,3
    jz      cambiar_modo
    mov     al,3                    ;o modo 3 para Color
cambiar_modo:
    int     10h                     ;Cambia el modo de video
abrir_ventana:
    call    abrir_ventana_editor    ;Abre ventana del Editor
    cmp     adaptador,1
    jne     no_cursor
    call    activar_CGA             ;Habilita salida de video
no_cursor:                              ;en la CGA
    call    ocultar_cursor          ;Oculta el cursor
    ret
VIDEO_PARAM ENDP


;
;                       RECUPERAR_VIDEO_PARAM
; Restablece los parámetros de video tal como estaban al
; inicio del programa.
;

RECUPERAR_VIDEO_PARAM       PROC    NEAR

; Si es necesario cambia al modo de video inicial.

    test    cs:modo_erroneo,1       ;¿ El modo de video fue
                                    ;modificado ?
    je      ck_cga                  ;No, continúe
    and     cs:modo_erroneo,0       ;Si, entonces
    xor     ah,ah                   ;recupera el modo de
    mov     al,modo_de_video        ;video inicial
    mov     bl,pagina_de_video
    int     10h
ck_cga:
    cmp     adaptador,1             ;Si existe una CGA,
    jne     cerrar_ventana
    call    desactivar_CGA          ;Desabilita la salida de
cerrar_ventana:                     ;video
    call    recuperar_video         ;Recupera el contenido de
    cmp     adaptador,1             ;la pantalla
    jne     actualizar_cursor
    call    activar_CGA             ;Habilita la salida de
                                    ;video
; Recupera la forma y la posición del cursor.

actualizar_cursor:
    mov     ah,2                    ;Fija el cursor en su
    mov     bh,pagina_de_video      ;posicion inicial
    mov     dx,pscion_del_cursor
    int     10h

; Muestra el cursor.
```

```
        call    activar_cursor          ;Activa el cursor
        ret
RECUPERAR_VIDEO_PARAM       ENDP


;
;   GUARDAR_VIDEO
;   Guarda el contenido del área de video donde se colocarán
;   las ventanas del EDITOR, DIRECTORIO, MENU, CONFIGURACION
;   y TELEX.
;

GUARDAR_VIDEO       PROC    NEAR
        mov     ds,sgmnto_de_video      ;DS:SI a la memoria de
        assume  ds:nothing              ;video
        xor     si,si                   ;Línea 0, Columna 0
        push    cs
        pop     es
        assume  es:code                 ;ES:DI al área reservada
        lea     di,area_para_el_video   ;para almacenar contenido
                                        ;de la pantalla.
        call    guardar_recuperar_video ;Guarda el contenido
        push    cs
        pop     ds                      ;Actualiza DS al segmento
        assume  ds:code                 ;del programa
        ret
GUARDAR_VIDEO       ENDP

;
;   RECUPERAR_VIDEO
;   Recupera el área de video modificada.
;

RECUPERAR_VIDEO     PROC    NEAR
        mov     es,sgmnto_de_video      ;ES:DI hacia la memoria
        assume  es:nothing              ;de video
        xor     di,di                   ;Línea 0, columna 0
        lea     si,area_para_el_video   ;DS:SI al área reservada.
        call    guardar_recuperar_video ;Recupera el contenido de
        ret                             ;la pantalla
RECUPERAR_VIDEO     ENDP

;
;   GUARDAR_RECUPERAR_VIDEO
;   Almacena o recupera el contenido de la pantalla.
;

GUARDAR_RECUPERAR_VIDEO     PROC    NEAR
        mov     cx,25                   ;25 líneas por almacenar
        cld                             ;o recuperar
```

```
guardar1:                                   ;Guarda el contador de
    push    cx                              ;líneas
    mov     cx,80                           ;80 caracteres por línea
    rep     movsw                           ;Transfiere una línea
    pop     cx                              ;recupera contador de
    loop    guardar1                        ;líneas y se realiza un
    ret                                     ;lazo hasta que todas las
                                            ;líneas hayan sido
GUARDAR_RECUPERAR_VIDEO     ENDP            ;transferidas.
```

```
;
;   ABRIR_VENTANA_EDITOR
;   Forma la ventana del  Editor  de Textos en la memoria de
;   video.
;
```

```
ABRIR_VENTANA_EDITOR        PROC    NEAR
    mov     es,sgmnto_de_video      ;ES:DI hacia la memoria
    assume  es:nothing              ;de video
    mov     bx,direc_video          ;Inicio de la ventana del
                                    ;EDITOR.
    mov     dh,80                   ;DH:80 Columnas
    mov     dl,12                   ;DL:12 Líneas
    mov     al,at_brd_txt           ;Atributos:
                                    ;Monoc-70h, Color-1Eh
    call    borde_ventana           ;Borde de la ventana
    call    info                    ;Información Superior
    call    fecha_y_hora            ;Fecha y hora actual
    mov     si,menu                 ;Menú de opciones en la
    call    mostrar_menu            ;última línea.
    ret
ABRIR_VENTANA_EDITOR        ENDP
```

```
;
;   INFO
;   Actualiza en la pantalla el contenido de la primera línea
;   del Editor de la siguiente forma:
;   TEXTO: [sendero][\]nombre_del_texto     Línea:      Col:
;
```

```
INFO        PROC    NEAR
    lea     si,menu2                ;Información de la línea
INFO1:
    mov     es,sgmnto_de_video      ;ES:DI a la memoria de
    assume  es:nothing              ;video.
    mov     di,direc_video
    add     di,2
    cld
    mov     cx,78                   ;78 caracteres mostrar
    mov     bh,at_info_1            ;Mono-70h, Color-3fh
    mov     al,at_info_2
```

```
        mov     atributo,al
        call    mostrar_linea
        ret
INFO    ENDP


;
;                          EDITOR
;          Permite que el texto sea editado y modificado.
;

EDITOR  PROC    NEAR
        lea     ax,inicio_area_de_textos ;Posición inicial del
                                         ;área de textos
        mov     cursor_texto,ax          ;Puntero del cursor de
                                         ;textos
        mov     inicio_pagina,ax         ;Inicio de la página del
edit1:                                   ;texto
        call    linea_columna            ;Muestra Línea/Columna de
                                         ;la posición del cursor
                                         ;de textos.
        mov     si,inicio_pagina         ;Inicio área de textos/
                                         ;área ventana de video
        call    pagina                   ;Muestra 12*78 caracteres
                                         ;del texto

; A partir de este momento, EDITOR está listo para ejecutar
; una de las opciones mostradas en la línea del menú.

edit2:                                   ;Espera por el ingreso
        call    tomar_tecla              ;desde el teclado
        cmp     al,ESC                   ;¿ ESC ?
        jz      fin_editor               ;Si, salir del EDITOR
        cmp     al,16h                   ;¿ Secuencia ^V ?
                                         ;Modo Conversacional
        jne     edit3                    ;No, chequear opciones
        call    MODO_CONVERSACIONAL      ;Si, llamar a la routina
        jmp     short edit1              ;de conversación
edit3:
        call    opciones_del_telex       ;Chequea opciones que
        jc      edit2                    ;ofrece el Telex
        cmp     al,1Fh                   ;¿ ASCII 32 o mayor ?
        ja      edit4                    ;Si, almacene caracter
        push    ax                       ;No, guarda AX
        mov     cx,27                    ;27 opciones del Editor
        call    ck_menu                  ;Chequea opciones de menú
        pop     ax                       ;Recupera AX
        je      edit1                    ;¿Alguna opción del menu?
                                         ;Si, entonces actualize
        or      al,al                    ;los cambios
        je      edit2                    ;Si AL=0 ignore chequeo
                                         ;de tildes
        call    tildes                   ;No, chequear tildes y
                                         ;eñe's
```

```
    jc      edit2                   ;¿Se ingresó tildes/eñe?
                                    ;No, ignore tecla
edit4:
    mov     si,cursor_texto         ;Si, entonces..
    mov     byte ptr ds:[si],al     ;almacena el caracter,
    call    avz_cursor              ;avanza a la siguiente
    jmp     short edit1             ;posición de memoria y
fin_editor:                         ;repite la secuencia
    ret

EDITOR      ENDP


;
;                          CK_MENU
;
;  Subroutina interpretadora del ingreso desde  el teclado.
;  Chequea si el código del caracter ingresado pertenece a
;  alguna opción del menú o si se está realizando una acción
;  de movimiento del cursor. Esta subroutina invoca a otras
;  dependiendo del  código de la tecla presionada.
;

CK_MENU     PROC
    push    cs                      ;Fija ES al segmento del
    pop     es                      ;programa.
    assume  es:code
    cld                             ;Para autoincremento
    lea     di,teclas_editor        ;DI en el área de códigos
                                    ;reservados
    mov     al,ah                   ;AH contiene la posición
                                    ;de la tecla presionada
    mov     bx,cx                   ;Guarda para después
    repne   scasb                   ;realiza comparación
    je      llamar_opcion           ;Sale con ZF=0 si no se
    ret                             ;encontró opción alguna
llamar_opcion:
    sub     bx,cx                   ;Calcula la posición de
    shl     bx,1                    ;la subroutina de opción.

; Ejecuta la subroutina de opción. ZF=1  indicará  que alguna
; subroutina fue invocada.

    call    word ptr cs:[bx]+subroutinas_del_editor-2
    xor     al,al                   ;ZF=1
    ret
CK_MENU     ENDP


;
;                          TILDES
;  Subroutina para  chequear  la  secuencia  CTRL-vocal-n-m
;  para formar las vocales con tildes y las eñe´s.
;
```

```
TILDES      PROC    NEAR
    cld
    lea     di,teclas_de_vocales    ;DI en códigos de
                                    ;CTRL-vocal-n-m.
    mov     cx,7                    ;7 comparaciones:
    mov     bx,cx                   ;(5 vocales, la ñ y la Ñ)
    repne   scasb                   ;Realiza la comparación.
    jne     no_tildes               ;¿ Alguna de ellas ?
                                    ;No - salir con CY=1
    sub     bx,cx                   ;Si
    lea     si,vocales_tildadas     ;Fijar SI en el área de
                                    ;códigos de letras
    mov     al,[si+bx-1]            ;y colocar valor en AL
    clc                             ;CY=0 para indicar el
    ret                             ;cambio.
no_tildes:
    stc                             ;CY=1 indica que no se ha
    ret                             ;detectado la secuencia
TILDES   ENDP                       ;CTRL-vocal-n-m.

;
;   Las  subroutinas siguientes realizan el movimiento  del
;   cursor del texto.
;

AVZ_CURSOR:                         ;Cursor una posición a la
    mov     cx,1                    ;derecha.
    jmp     short nueva_posicion
RET_CURSOR:                         ;Una posición hacia la
    mov     cx,-1                   ;izquierda.
    jmp     short nueva_posicion
BAJAR_CURSOR:
    mov     cx,78                   ;Una posición hacia abajo
    jmp     short nueva_posicion
SUBIR_CURSOR:
    mov     cx,-78                  ;Una posición hacia
nueva_posicion:                     ;arriba.
    mov     si,cursor_texto         ;Cursor del texto en SI
    add     si,cx                   ;Suma a SI, CX posiciones
cmp si,offset inicio_area_de_textos ;¿ SI fuera del área de
                                    ;textos? (pos. inferior)
    jae     ck_fin_txt              ;No, ver fin del texto
    ret                             ;Si, ignore cambio
ck_fin_txt:
  cmp  si,offset fin_area_de_textos ;¿ SI fuera del área de
                                    ;textos? (pos. superior)
    jb      ck_ini_pg               ;No, chequear inicio de
                                    ;página
    ret                             ;Si, ignore cambio
ck_ini_pg:
    cmp     si,inicio_pagina        ;¿ SI fuera de página ?
                                    ;(Inferior)
    jae     ck_fin_pg               ;No, chequear fin de
                                    ;página
```

```
    sub     inicio_pagina,78        ;Si, bajar puntero de
    jmp     short avance            ;página.
ck_fin_pg:
    mov     bx,inicio_pagina
    add     bx,12*78
    cmp     si,bx                   ;¿ SI fuera de página ?
                                    ;(Superior)
    jb      avance                  ;No, incremente el cursor
                                    ;de texto según SI
    add     inicio_pagina,78        ;Si, subir puntero de
avance:                             ;página
    mov     cursor_texto,si         ;Avanza cursor de memoria
    ret                             ;y retorna
```

;
; **INICIO_DE_LINEA:** Posición inicial de la línea
;

```
INICIO_DE_LINEA:
    mov     ax,cursor_texto         ;Calcula número de líneas
    sub     ax,inicio_pagina        ;desde la posición inicial
    xor     dx,dx                   ;de la ventana hasta la
    mov     bx,78                   ;posición que contiene el
    div     bx                      ;cursor.
    sub     cursor_texto,dx         ;Luego resta al cursor el
    mov     cx,77                   ;residuo obtenido en DX
    sub     cx,dx                   ;CX contiene lo que falte
    ret                             ;completar la línea.
```

;
; **INTRO:** Posición inicial de la siguiente línea
;

```
INTRO:
    call    inicio_de_linea         ;Cursor a inicio de línea
    jmp    short bajar_cursor       ;luego, baja a la línea
                                    ;siguiente
```

;
; **TABULADOR:** Movimiento del cursor. Hasta 8 posiciones
;

```
TAB:
    call    inicio_de_linea
    mov     ax,dx                   ;Cursor a inicio de línea
    cmp     ax,70                   ;Si el cursor está casi
                                    ;al final de la línea,
    jae     bajar_cursor            ;se va a la siguiente
                                    ;línea.
```

```
    add     cursor_texto,dx         ;Caso contrario, divide
    mov     ax,dx                   ;la línea del cursor para
    add     al,8                    ;un valor de 8.
    mov     cx,8
    div     cl
    sub     cl,ah
    add     cursor_texto,cx         ;Suma al cursor el residuo
    ret                             ;obtenido en CX, necesario
                                    ;para completar un TAB.

;
;   FIN_DE_LINEA: Final de Línea
;

FIN_DE_LINEA:

    call    inicio_de_linea         ;Cursor inicio de línea
    mov     di,cursor_texto         ;luego se le suma 77 para
    add     di,77                   ;posicionarlo al final de
                                    ;la línea
    mov     al,' '                  ;A partir de este lugar,
    mov     cx,77                   ;se comienza a restar el
    std                             ;cursor cada vez que se
                                    ;encuentre en la posición
el2:                                ;de un espacio en blanco.
    scasb                           ;Finalmente, el cursor
    jne     el3                     ;queda en la siguiente
    loop    el2                     ;posición del último
    ret                             ;caracter diferente de un
el3:                                ;espacio en blanco,
    cmp     cx,77                   ;existente en la línea.
    je      el4
    inc     cx
el4:
    add     cursor_texto,cx
    ret

;
;   BCKSPC: Retroceder y borrar
;

BCKSPC:

    call    ret_cursor              ;Primero retrocede el
    mov     byte ptr ds:[si],' '    ;cursor y luego borra el
    ret                             ;caracter

;
;   SUBIR_PG: Página hacia arriba (PG UP)
;
```

```
SUBIR_PG:
    mov     si,inicio_pagina        ;Si el cursor está en la
    cmp     si,cursor_texto         ;posición inicial del
    je      up1                     ;texto se ignora el
    mov     cursor_texto,si         ;pedido de subida.
    ret                             ;Caso contrario,

up1:
    call    num_linea               ;Primero se chequea en
    cmp     al,11                   ;qué línea de la pantalla
                                    ;se encuentra el cursor.
    jbe     pg2                     ;Si el cursor está en la
    mov     al,11                   ;posición inicial de la

pg2:
    xor      dx,dx                  ;primera linea,  se  le
                                    ;resta el valor de 11
                                    ;líneas.
    mul     bx                      ;Caso contrario, se fija
    sub     inicio_pagina,ax        ;el cursor en la posición
    sub     cursor_texto,ax         ;inicial de la primera
    ret                             ;línea

;
;   BAJAR_PG: Página hacia abajo (PG DN)
;

BAJAR_PG:
    mov     si,inicio_pagina        ;Si el cursor está al
    add     si,11*78                ;inicio de la última
    cmp     si,cursor_texto         ;linea de la ventana, se
                                    ;le suma el valor de 11
                                    ;líneas.
    je      dn1                     ;Caso contrario, se fija
    mov     cursor_texto,si         ;el cursor en ésta
    ret                             ;posición.

dn1:
    mov     cx,11                   ;Además, si el cursor
    mov     si,cursor_texto         ;está en la última línea
                                    ;disponible para
                                    ;almacenamiento de textos,
dn2:
    cmp     si,offset fin_area_de_textos-78 ;ninguna acción
    je      dn3                     ;se realiza.
    add     si,78
    loop    dn2

dn3:
    mov     cursor_texto,si         ;Si está cerca de la
    sub     si,11*78                ;última línea,se fija el
    mov     inicio_pagina,si        ;en esta última línea.
    ret
```

```
;
;                          INICIO_PG
;     Posiciona el cursor al inicio del texto (CTRL PG UP)
;

INICIO_PG:

     lea     si,inicio_area_de_textos ;SI contiene la posición
     jmp      short pg_hm_fin         ;donde queremos colocar
                                      ;el cursor.

;
;                          FIN_PG
;     Posiciona el cursor al final del texto (CTRL PG DN)
;

FIN_PG:

    lea  si,fin_area_de_textos-78*12 ;Se fija el cursor en la
                                     ;posición de 11 líneas
pg_hm_fin:                           ;antes del final de texto
     mov      cursor_texto,si        ;Actualiza el cursor e
     mov      inicio_pagina,si       ;inicio de página.
     call     pagina
     ret

;
;                       BORRAR_CARACTER
;          Borra el caracter de la posición del cursor
;

BORRAR_CARACTER:

     push     cursor_texto           ;Calcula en CX lo que
     call     inicio_de_linea        ;falta para completar la
     pop      cursor_texto           ;línea a partir de la
                                     ;posición del cursor.
     mov      di,cursor_texto        ;Si el cursor está en la
     mov      si,di                  ;posición del caracter
     inc      si                     ;la última línea, lo
     jcxz     dch                    ;borra.

     cld                             ;Caso contrario, mueve
                                     ;hacia la izquierda todos
     rep      movsb                  ;los caracteres de la
                                     ;línea que se encuentran

dch:
     mov      byte ptr [si-1],' '    ;a partir de la posición
     ret                             ;del cursor.
```

```
;
;                          BORRAR_LINEA
;                 Borra la línea de la posición del cursor
;

BORRAR_LINEA:

    call    inicio_de_linea          ;Fija el cursor al inicio
                                     ;de la línea.
    mov     di,cursor_texto          ;Luego mueve todo el
    mov     si,cursor_texto          ;texto que se encuentra a
    add     si,78                    ;partir de la línea
                                     ;siguiente, una línea
                                     ;hacia arriba.
    lea     cx,fin_area_de_textos    ;CX contiene el número de
    sub     cx,cursor_texto          ;líneas a mover.
    cld
    rep     movsb                    ;Mueve las lineas
    ret

;
;                          INS_CARACTER
;              Inserta un caracter en la posición del cursor
;

INS_CARACTER:
    push    cursor_texto             ;Esta subroutina realiza
    call    inicio_de_linea          ;lo contrario a DEL_CHR;
                                     ;mueve todos los
    mov     di,cursor_texto          ;caracteres existentes en
    add     di,77                    ;la línea del cursor, una
    mov     si,di                    ;posición a la derecha,
    dec     si                       ;dejando un espacio en
    pop     cursor_texto             ;a la derecha del cursor.
    jcxz    ich                      ;El último caracter de la
    std                              ;línea siempre se perderá
    rep     movsb
ich:
    mov     byte ptr [si+1],' '
    ret

;
;                          INS_LINEA
;              Inserta una línea en la posición del cursor
;

INS_LINEA:
    call    inicio_de_linea          ;Inserta una línea en la
    lea  si,fin_area_de_textos - 80  ;línea anterior de la
    lea  di,fin_area_de_textos - 2   ;línea que contiene el
    cmp     si,cursor_texto          ;cursor.
```

```
        jae     ins
        ret
ins:
        mov     cx,si                   ;Primero mueve todo el
        sub     cx,cursor_texto         ;texto una línea hacia
        shr     cx,1                    ;abajo, a partir de la
        inc     cx                      ;línea que contiene el
        std                             ;cursor.
        rep     movsw
        cld
        mov     di,cursor_texto         ;Luego deja esta línea
        mov     ax,2020h                ;con espacios en blanco
        mov     cx,39                   ;y con el cursor en la
        rep     stosw                   ;posición del inicio de
        ret                             ;la línea.

;
;                              PAGINA
;  Muestra en la ventana del EDITOR 12 líneas x 78 columnas
;  de texto.
;

PAGINA  PROC    NEAR
        mov     es,sgmnto_de_video      ;ES:DI en la posición
        assume  es:nothing              ;inicial de video de la
        mov     di,direc_video          ;ventana del EDITOR.
        add     di,162
        mov     cx,12                   ;12 líneas de texto
        cld
        mov     bh,attr_de_texto

pgn1:
        push    cx                     ;Guarda contador de líneas
        mov     cx,78                   ;78 caracteres por línea.

pgn2:
        lodsb                           ;Toma caracter del texto,
        mov     bl,al
        call    escribir_palabra        ;y lo envia a la memoria
        mov     dx,si                   ;de video.
        dec     dx                      ;Si el caracter enviado
                                        ;está en la posición que
        cmp     dx,cursor_texto         ;que contiene el cursor,
        jne     pgn3                    ;se le cambia su atributo
        mov     byte ptr es:[di-1],70h  ;de color para simular la
                                        ;existencia de un cursor
pgn3:                                   ;en la pantalla.
        loop    pgn2                    ;Lazo para enviar línea
        add     di,4                    ;Posición de la línea
        pop     cx                      ;siguiente por mostrar.
        loop    pgn1                    ;Continúa con la línea
        ret                             ;siguiente.
PAGINA  ENDP
```

```
;
;                              LINEA-COLUMNA
;  Muestra el número de línea y columna en la que se
;  encuentra el cursor del texto.
;

LINEA_COLUMNA        PROC
     mov      es,sgmnto_de_video        ;ES:DI en el área de
     mov      di,direc_video            ;video donde se colocará
     add      di,67*2                   ;el valor de la línea
     call     num_linea                 ;Calcula línea y columna
                                        ;del cursor
     push     dx                        ;Guarda columna
     cld
     call     mostrar_lc                ;Muestra valor de línea
                                        ;del cursor
     add      di,6*2                    ;Posición para mostrar
                                        ;columna
     pop      dx                        ;Recupera columna
     mov      al,dl
     call     mostrar_lc                ;Muestra valor de columna
     ret                                ;del cursor.
LINEA_COLUMNA        ENDP

;
;                              NUM_LINEA
;  Entrega en AX:DX el número de la línea en la que se
;  encuentra el cursor del texto.
;

NUM_LINEA:
     mov      ax,cursor_texto
linea:
     lea    si,inicio_area_de_textos  ;Resta la posición del
     sub      ax,si                   ;cursor del inicio texto.
     mov      bx,78                   ;y luego divide el
     xor      dx,dx                   ;resultado para 78.
     div      bx                      ;AX contiene la línea
     ret                              ;y DX la columna

;
;                              MOSTRAR_LC
;  Toma el valor de la línea o columna del cursor, lo
;  convierte a un valor ASCII y lo muestra en la primera
;  línea de la ventana del EDITOR.
;

MOSTRAR_LC:
     mov      bl,0010                 ;Convierte valor dado en
     xor      ah,ah                   ;AL a un valor ASCII
```

```
    div     bl
    or      al,30h                  ;y guarda valor MSB en BL
    mov     bl,al                   ;para ser mostrado
    mov     bh,lincol_attr          ;Mono-70h, Color-31h
    push    ax                      ;Antes de enviar, guarda
                                    ;valor LSB.
    call    escribir_palabra        ;Muestra valor MSB,
    pop     ax                      ;Recupera valor LSB,
    or      ah,30h                  ;Lo convierte a valor
    mov     bl,ah                   ;ASCII,
    call    escribir_palabra        ;y lo muestra.
    ret

;
;       GRABAR: Comprime el texto y lo graba
;

GRABAR      PROC    NEAR
    call    final_de_texto          ;¿ Existe algún texto por
                                    ;grabar ?
    jc      salir_grabar            ;No, ignora pedido
    call    context                 ;Si, comprime el texto
    call    grabar_texto            ;y llama a la subroutina
salir_grabar:                       ;que debe tratar de
    ret                             ;grabar el texto.
GRABAR      ENDP

;
;                       CONTEXT
;  Comprime el texto. Esta subroutina evita en lo posible
;  los espacios en blanco existentes al final de cada línea
;  insertando en su lugar caracteres de Retorno del Carro
;  CR:Carriage Return y de Avance de Línea LF:Line Feed.
;

CONTEXT     PROC    NEAR
    cld                             ;Ajusta el número de
    call    ajtop                   ;líneas del texto
    mov     si,fin_del_texto        ;Fija SI al final del
    push    cs                      ;texto.
    pop     es                      ;ES al seg. del programa.
lea di,nombres_de_archivos+7999     ;DI al final del área
                                    ;donde se colocará el
ct1:                                ;texto comprimido
    mov     cx,78                   ;Longitud máxima de
    dec     di                      ;línea.
ct2: cmp    byte ptr [si],' '       ;¿ Fin de línea ?
    jne     ct3                     ;Si, inserte un CR-LF
    dec     si                      ;No, posición anterior
    loop    ct2
ct3:
```

```
    mov     word ptr [di],0A0Dh     ;Inserta un CR-LF
    dec     di                      ;Posición anterior
                                    ;¿ Linea vacia ?
    jcxz    ct5                     ;Si, chequea si todo el
                                    ;texto se ha comprimido
    std                             ;No, entonces...
    rep     movsb                   ;Transfiere caracteres de
ct5: lea    ax,inicio_area_de_textos ;la línea.
    cmp     si,ax                   ;¿SI al inicio del texto?
    ja      ct1                     ;No, continúe
    inc     di                      ;Si, entonces...
lea ax,nombres_de_archivos+8000     ;Calcula el número de
    sub     ax,di                   ;bytes que contiene el
    mov     num_bytes,ax'           ;texto comprimido,
    mov     dir_Grb_Imp_Env,di      ;y la posición inicial
    ret                             ;del mismo.
CONTEXT     ENDP


;
;                          AJTOP
;  Fija el puntero de final del texto al inicio de su línea
;  siguiente.
;

AJTOP       PROC    NEAR
    mov     ax,fin_del_texto        ;Al grabarse el texto,
    call    linea                   ;siempre contendrá un
    sub     fin_del_texto,dx        ;número entero de líneas
    add     fin_del_texto,77
    ret
AJTOP       ENDP


;
;  GRABAR_TEXTO: Graba el texto en memoria hacia el disco
;

GRABAR_TEXTO        PROC    NEAR

; Muestra el nombre del texto a grabar en la línea del menú

    lea     si, menu2+9             ;Copia el nombre del
    lea     di,menu5+17             ;texto que se encuentra
    mov      cx,35                  ;en la primera de la
                                    ;ventana, después del
                                    ;mensaje 'Grabar archivo'
    cld                             ;para ser mostrado en la
    rep     movsb                   ;línea de menú.
    lea     si,menu5                ;Muestra el mensaje:
    call    mostrar_menu            ;'Grabar archivo:
                                    ;[nombre del texto]_'
    lea     si, menu2+9             ;Asi mismo, copia el
```

```
    lea     di,sendero              ;nombre del texto en la
                                    ;cadena a partir de PATH
    xor     cl,cl                   ;Contador para ruta y
grb0:                               ;nombre del texto
    cmp  byte ptr [si],' '          ;¿Fin nombre del texto ?
    jz      grb1                    ;Si, cursor
    movsb                           ;No, copiar caracter
    inc     cl                      ;Incrementar contador
    jmp     short grb0              ;Continuar copiando
grb1:                               ;Calcula posición final
    mov     dl,16                   ;donde se colocará el
    add     dl,cl                   ;cursor cuando se muestre
    mov     dh,13h                  ;el nombre del texto en
                                    ;la línea del menú.
    call    leer_cadena             ;Lee línea de caracteres
                                    ;(Para modificación del
                                    ;nombre del texto).
    or      cl,cl                   ;¿ Salida con ESC ?
    je      salir_archivo           ;Si, ignorar grabado
    mov  byte ptr es:[di],0         ;No, colocar un '0' al
                                    ;final del nombre del
                                    ;texto para formar una
                                    ;cadena ASCIIZ.

; Abre el archivo y graba el texto contenido en el Editor.

    lea     dx,sendero              ;Cadena ASCIIZ contiene
                                    ;el nombre del texto
    mov     ah,3ch                  ;Función del DOS para
                                    ;abrir/crear un archivo
    xor     cx,cx                   ;atributos normales
    int     21h                     ;abre/crea el archivo
    jc      grabar_texto            ;Si ocurrió un error en
                                    ;la apertura, salta a
                                    ;tomar otro nombre para
                                    ;el texto
    mov     puntero_del_archivo,ax  ;Guarda el puntero del
    mov     bx,ax                   ;fichero y lo copia en BX.
    mov     ah,40h                  ;Función del DOS para
                                    ;escribir en el fichero
    mov  dx,dir_Grb_Imp_Env         ;Dirección desde donde se
                                    ;grabará el texto
    mov     cx,num_bytes            ;Número de bytes a grabar
    int     21h                     ;Graba el texto
    lea     si, menu7               ;Mensaje:
                                    ;'Error grabando archivo'
    jc      error_de_escritura      ;Si ocurrió un error,
                                    ;muestra un mensaje de
                                    ;error de escritura.
    cmp     ax,cx                   ;¿ El texto fue grabado
                                    ;completamente ?
    jb      disco_lleno             ;No, indicar que no hay
                                    ;espacio suficiente.
                                    ;Si, entonces..
    lea     si,sendero
```

```
    call    guardar_nuevo_nombre    ;Guarda el nuevo nombre
                                    ;del texto
    call    info                    ;lo muestra en la primera
    jmp     short cerrar_archivo    ;línea de la ventana y
                                    ;cierra el archivo

; Error de escritura o espacio insuficiente en el disco.

disco_lleno:
    lea     si, menu8               ;Mensaje:
error_de_escritura:                 ;'Espacio insuficiente..'
    call    mostrar_menu            ;Muestra el mensaje.
    call    tomar_tecla             ;Espera que se presione
cerrar_archivo:                     ;cualquier tecla.
    mov     bx,puntero_del_archivo  ;Recupera puntero del
                                    ;fichero,
    mov     ah,3Eh                  ;y cierra el archivo
    int     21h
salir_archivo:                      ;Finalmente,
    mov     si,menu                 ;Muestra el menú
    call    mostrar_menu            ;principal..
    ret                             ;y sale
GRABAR_TEXTO        ENDP

;
;   LEER_ARCHIVO: Lee un archivo desde el disco.
;

LEER_ARCHIVO        PROC    NEAR
    lea     si, menu6               ;Muestra el mensaje:
    call    mostrar_menu            ;'Leer archivo: '
    lea     di,sendero              ;Cadena que contiene el
                                    ;nombre del archivo
    mov     dx,130Eh                ;Posición del cursor
    xor     cl,cl                   ;Contador de caracteres
    call    leer_cadena             ;Lee el nombre del
                                    ;archivo
    or      cl,cl                   ;¿ Salida con ESC ?
    je      salir_archivo           ;Si, ignore pedido
    mov     byte ptr es:[di],0      ;Completa cadena ASCIIZ
    mov     ax,3D00h                ;Función para abrir un
                                    ;archivo
    lea     dx,sendero              ;Nombre del archivo
    int     21h                     ;Abre el archivo
    jnc     leer1                   ;¿Algún error de lectura?
                                    ;No, continúe
    lea     si, menu9               ;Si, entonces muestra el
                                    ;mensaje:
    call    mostrar_menu            ;'Archivo no encontrado'
    call    tomar_tecla             ;Espera que se presione
                                    ;una tecla,
    jmp     leer_archivo            ;y vuelve a tomar otro
leer1:                              ;nombre de archivo
```

```
    mov     bx,ax                       ;Puntero del archivo.
    mov     puntero_del_archivo,bx      ;Guarda el puntero.
    mov     ah,3Fh                      ;Función para leer del
                                        ;archivo
    mov     cx,1                        ;Número de bytes a leer
    lea     dx,nombres_de_archivos      ;Area donde se sitúan los
                                        ;datos leídos
    int     21h                         ;Lee un byte
    jc      cerrar_archivo              ;Retorna si ocurrió algún
                                        ;error de lectura
    mov     cx,7800-1                   ;Caso contrario lee hasta
                                        ;7800 bytes
    mov     ah,3fh                      ;Después de que la
                                        ;función se ha ejecutado,
    lea   dx,nombres_de_archivos+1      ;AX contiene el número
    int     21h                         ;real de bytes leídos.
    mov     fin_de_archivo,ax
    call    cerrar_archivo              ;Cierra el archivo.
    call    formar_texto                ;Convierte el archivo
                                        ;leído a formato de texto
    lea     si,sendero                  ;Copia el nombre del
                                        ;archivo leído en el
    call    guardar_nuevo_nombre        ;mensaje de la primera
                                        ;línea
    call    info                        ;Actualiza la primera
    ret                                 ;línea de la ventana y
LEER_ARCHIVO        ENDP                ;retorna.

;
;                    GUARDAR_NUEVO_NOMBRE
;  Copia el nombre del archivo grabado o leído desde el
;  disco, en el área del mensaje que se muestra en la
;  primera línea de la ventana del EDITOR.
;

GUARDAR_NUEVO_NOMBRE        PROC    NEAR
    lea     di,menu2+9                  ;Mensaje
    mov     cx,36                       ;Hasta 36 caracteres a
nombre_de_texto:                        ;copiar.
    cmp     byte ptr [si],0             ;¿Fin nombre del texto ?
    jz      mt2                         ;Si, borrar caracteres
                                        ;de la derecha
    movsb                               ;No, toma el siguiente
                                        ;caracter y lo almacena
    loop    nombre_de_texto             ;continúa.
    ret
mt2:
    push    di                          ;Si el número de bytes
    mov     al,' '                      ;que contiene el nombre
    rep     stosb                       ;del archivo es inferior
    pop      di                         ;a 36, se borran todos
                                        ;los caracteres que
                                        ;existan a la derecha de
```

```
    ret                                     ;la posición de MENU2+DI
GUARDAR_NUEVO_NOMBRE        ENDP


;
;                          FORMAR_TEXTO
;  Transforma el archivo tomado desde el disco a un
;  formato de texto. Anula los caracteres de CR, LF y TAB.
;


FORMAR_TEXTO        PROC

    call    borrar_area_txt             ;Borra el área donde se
                                        ;formará el texto.
    lea     si, nombres_de_archivos     ;Posición inicial del
                                        ;archivo leído del disco
    lea   di,inicio_area_de_textos      ;Posición inicial del
    cld                                 ;TEXTO a formar.
    mov     cx,fin_de_archivo           ;Longitud máxima del
                                        ;texto
    xor     bx,bx                       ;Contador de caracteres
form1:                                  ;por línea
    lodsb                               ;Toma caracter
    cmp     al,ENTER                    ;¿ Código de INTRO ?
    jz      cr                          ;Si, realizar un retorno
                                        ;de carro
    cmp     al,10                       ;¿ Código de LINE FEED ?
    jz      lf                          ;Si, realizar un avance
                                        ;e línea
    cmp     al,09                       ;¿ Código de TABULADOR ?
    jz      tabdor                      ;Si, realizar un TAB
    stosb                               ;Caso contrario, almacena
                                        ;caracter e
    inc     bl                          ;Incrementa contador de
form2:                                  ;caracteres en la línea
    loop    form1                       ;Si  CX<>0,  toma  el
                                        ;siguiente caracter
    ret                                 ;Caso contrario, retorna

; Ejecuta una acción de avance de línea.

lf:
    xor     ax,ax                       ;AX=0
lf1:
    add     ax,78                       ;Avanza una línea
    cmp     ax,num_bytes                ;¿ AX < Longitud de
                                        ;caracteres de línea
                                        ;anterior ?
    jb      lf1                         ;Si, realize otro avance
    mov     num_bytes,0                 ;de línea.
    add     di,ax                       ;DI avanza una línea
  cmp  di,offset fin_area_de_textos     ;¿DI al final del texto ?
    jb      form2                       ;No, continue.
    ret                                 ;Si, salir
```

```
; Ejecuta una acción de retorno del carro.

cr:                                     ;Resta a DI el número de
    sub     di,bx                       ;caracteres que están
                                        ;presentes en la línea
    mov     num_bytes,bx                ;Guarda el número de
    xor     bl,bl                       ;caracteres de la línea.
    jmp     short form2                 ;Continúa.

; Ejecuta una acción de tabulador.

tabdor:
    call    tabulador                   ;Subroutina de TAB
    jmp     short form2'                ;Tomar siguiente caracter
    ret
FORMAR_TEXTO        ENDP


TABULADOR   PROC
    cmp     bl,70                       ;Ultima posición
                                        ;disponible del TAB
    jb      tabdor1                     ;No, realice un TAB
    sub     di,bx                       ;Si, avanza diréctamente
    add     di,78                       ;a la primera posición de
    xor     bl,bl                       ;la siguiente línea.
    ret
tabdor1:

    xor     ah,ah
    mov     al,bl                       ;AX=Número de caracteres
    mov     dl,8                        ;en la línea.
    div     dl                          ;Divide la línea para
                                        ;valor del TAB (8)
    sub     dl,ah                       ;DL en la posición exacta
                                        ;del TAB anterior
    or      ah,ah                       ;¿ Posición inicial de la
                                        ;línea ?
    jne     tabdor2                     ;No, agrege lo que falte
                                        ;para completar un TAB.
    mov     dl,8                        ;Si, TAB de 8

tabdor2:
    add     bl,dl                       ;BL en la posición del
    xor     dh,dh                       ;TAB siguiente.
    add     di,dx                       ;Realiza lo mismo para DI
    ret                                 ;y sale
TABULADOR   ENDP


;
;                          LEER_CADENA
;   Lee  desde  el  teclado  una  cadena  de  hasta  35
;   caracteres.
;   CX contendrá el número de caracteres ingresados.
;
```

```
LEER_CADENA     PROC  NEAR

    push    cx
    call    fijar_cursor            ;Posiciona y muestra el
    pop     cx                      ;cursor según el valor
                                    ;dado en el registro  DX

; Espera por el ingreso desde el teclado.

lcad:
    call    tomar_tecla             ;Toma un caracter
    cmp     al,ENTER                ;¿ Tecla INTRO ?
    je      salir_leer              ;Si, fin de ingreso
    cmp     al,ESC                  ;¿ ESC ?
    jne     leer2                   ;No, procesar caracter
                                    ;ingresado
    xor     cl,cl                   ;Si, entonces ignora los
    jmp     short salir_leer        ;ingresos igualando a
                                    ;cero el contador de
                                    ;caracteres y sale.
leer2:
    cmp     al,8                    ;¿ Retroceder ?
    je      backspace               ;Si, entonces retroceder
                                    ;y borrar
    cmp     al,32                   ;¿ ASCII 32 o mayor ?
    jb      lcad                    ;No, ignore
    cmp     al,126                  ;¿ ASCII 126 o menor ?
    ja      lcad                    ;No, ignore
    cmp     cl,35                   ;¿ Existe espacio para
                                    ;otro caracter ?
    je      lcad                    ;No, entonces esperar por
                                    ;INTRO, ESC o Retroceder

 ;Procesa el caracter ingresado.

    stosb                           ;Deposita al caracter en
                                    ;la variable SENDERO
    push    cx                      ;Guarda contador de
                                    ;caracteres
    mov     ah,10                   ;Usa el BIOS para mostrar
    mov     cx,1                    ;el caracter ingresado.
    int     10h
    inc     dl                      ;Incrementa y avanza la
    mov     ah,2                    ;posición del cursor
    int     10h
    pop     cx                      ;Recupera e incrementa el
    inc     cl                      ;contador de caracteres
    jmp     lcad                    ;Toma más caracteres

 ;Realiza la función de retroceder y borrar.

backspace:
    or      cl,cl                   ;¿ Existe algún caracter
                                    ;por borrar ?
    je      lcad                    ;No, entonces ignore la
```

```
                                        ;tecla presionada.
    push    cx                          ;Si, guarda el contador
    dec     dl                          ;Retrocede el cursor una
    mov     ah,2                        ;posición.
    int     10h
    mov     ah,10
    mov     al,32                       ;Muestra un espacio en
    mov     cx,1                        ;blanco.
    int     10h
    pop     cx                          ;Recupera y decrementa el
    dec     cl                          ;contador.
    dec     di                          ;Decrementa puntero.
    jmp     lcad                        ;Salta a tomar caracteres
salir_leer:                             ;adicionales.
    mov     ah,1
    mov     ch,20h                      ;Oculta el cursor
    int     10h
    ret                                 ;y sale.

LEER_CADENA ENDP

;
;   IMPRIMIR: Imprime el texto
;

IMPRIMIR    PROC    NEAR

    call    final_de_texto              ;¿Existe algún texto para
                                        ;imprimir ?
    jc      fin_imprimir                ;No, ignore pedido.
    lea     si, menu15                  ;Si, muestra el mensaje:
    call    mostrar_menu                ;"Prepare impresora... "
    call    intro_esc                   ;Luego espera por INTRO o
                                        ;ESC.
    jc      fin_imprimir                ;¿ ESC ?, Salir
    call    context                     ;Comprime el Texto
    mov     di,dir_Grb_Imp_Env          ;Dirección desde donde se
prn1:                                   ;imprimirá.
    lea     si, menu10                  ;Muestra el mensaje:
    call    mostrar_menu                ;"Imprimiendo..[P] pausa"
prn2:
    call    chequear_impresora          ;Chequea la impresora
    jc      fin_imprimir                ;Sale si Cy=1
    call    tecla                       ;Chequea el teclado
    and     al,5Fh                      ;Convierte a mayúscula
    cmp     al,'P'                      ;¿ [P] Pausa ?
    jne     prn3                        ;No, continuar ...
    lea     si, menu12                  ;Si, entonces muestra el
                                        ;mensaje:
    call    mostrar_menu                ;"Presione INTRO para
                                        ;continuar o ESC para
                                        ;salir"
    call    intro_esc                   ;Espera por INTRO o ESC
```

```
    jc      fin_imprimir            ;ESC ? Sí, entonces salir
    jmp     short prn1              ;INTRO, continúa
 prn3:                              ;imprimiendo.
    xor     dx,dx                   ;Impresora 0
    xor     ah,ah                   ;Función para enviar un
                                    ;byte a la impresora
    mov     al,byte ptr cs:[di]     ;Toma el byte a imprimir
    int     17h                     ;Servicio de impresora
    inc     di                      ;Posición del siguiente
                                    ;caracter
    dec     num_bytes               ;¿Algún caracter más
                                    ;por imprimir ?
    jne     prn2                    ;Si, continúe
fin_imprimir:
    mov     si,menu                 ;No, entonces actualiza
    call    mostrar_menu            ;la línea de menú.
    ret
IMPRIMIR    ENDP


;
;                       CHEQUEAR_IMPRESORA
;  Chequea el estado de la impresora. A su salida CY informa
;  si la impresora está lista.
;

CHEQUEAR_IMPRESORA  PROC    NEAR

    mov     ah,02                   ;Función para obtener el
    mov     dx,00                   ;estado de la impresora
    int     17h                     ;Obtiene estado de la
                                    ;impresora 0
    test    ah,00101001b            ;¿ Impresora lista ?
    je      lista                   ;Si, indicarlo con CY=0
    lea     si, menu11              ;No, entonces muestra el
                                    ;mensaje:
    call    mostrar_menu            ;Error imprimiendo texto.
    call    intro_esc               ;y espera por el ingreso
                                    ;de INTRO o ESC.
    jnc     impresora_ok            ;INTRO ?, sale con CY=0
                                    ;ESC ?, Sale con CY=1,
                                    ;Indicando que el proceso
    ret                             ;de impresión debe ser
                                    ;anulado.
impresora_ok:

    lea     si, menu10              ;Deja el mensaje de
    call    mostrar_menu            ;impresión.
lista:
    clc                             ;Retorna con CY=0 para
                                    ;indicar que la impresora
    ret                             ;está lista para recibir
                                    ;algún caracter.
CHEQUEAR_IMPRESORA  ENDP
```

```
;
;                          FINAL_DE_TEXTO
;   Al salir, CY=0 indica que existe  algún texto disponible
;   para  ser  grabado,  impreso  o  enviado por la línea de
;   comunicación. CX contendrá el número de bytes del texto.
;

FINAL_DE_TEXTO      PROC    NEAR

     mov     cx,7800                    ;Chequea 7800 bytes
     mov     al,' '                     ;Espacio en blanco
     push    cs                         ;ES = CS
     pop     es
     lea  di,cs:fin_area_de_textos-1    ;Posición final del área
     std                                ;de textos.
     repe    scasb                      ;Chequea si toda el  área
                                        ;está vacia
     jz      no_texto                   ;Si es así, muestra un
                                        ;mensaje de error
 add cx,offset inicio_area_de_textos;Caso contrario, calcula
                                        ;la posición final del
                                        ;texto.
     mov     fin_del_texto,cx           ;CX = Número de bytes del
                                        ;texto.
     clc                                ;CY = 0 indica que existe
     ret                                ;algún texto.
no_texto:
     lea     si, menu14                 ;Envia el mensaje:
     call    mostrar_menu               ;"No existe texto alguno"
     call    parlante                   ;Suena el parlante
     call    tomar_tecla                ;Espera por el ingreso
                                        ;desde el teclado
     mov     si,menu                    ;Recupera el menú inicial
     call    mostrar_menu
     stc                                ;y retorna con CY = 1
     ret

FINAL_DE_TEXTO      ENDP


;
;   PARLANTE:  Subroutina para hacer sonar el parlante.
;


PARLANTE:
     mov     al,10110110b
     out     43h,al
     mov     ax,1000                    ;Selecciona frecuencia
     out     42h,al                     ;y envia un valor a
     mov     al,ah                      ;la vez
     out     42h,al
     in      al,61h                     ;Suena el parlante
     mov     ah,al
     or      al,3
```

```
    out     61h,al
    mov     cx,duracion             ;CX controla la duración

retardo:
    loop    retardo
    mov     al,ah                   ;Apaga el parlante
    out     61h,al
    ret

;
;   PULSACION
;   Subroutina que hace sonar el parlante cada vez que una
;   tecla es presionada, estando en el modo Conversacional
;   del TELEX.
;

PULSACION:

    mov     bx,ax                   ;Guarda AX
    mov     duracion,2000           ;Duración del sonido
    call    parlante                ;suena el parlante
    mov     duracion,7FFFh
    mov     ax,bx                   ;recupera AX
    ret

;
;   TOMAR_TECLA
;   Espera por el ingreso desde teclado y retorna en el
;   registro AX el código (ASCII) y la posición de la tecla
;   presionada.
;

TOMAR_TECLA PROC    NEAR

    mov     ah,1                    ;Chequea el área del
    int     16h                     ;teclado

    jne     tecla_lista             ;Salta si el área tiene
                                    ;algún código,
    int     28h                     ;Caso contrario, se usa
                                    ;una INT 28h
    jmp     short tomar_tecla       ;e intenta de nuevo

tecla_lista:

    mov     ah,0                    ;Toma el código de la
                                    ;tecla desde el área y
    int     16h                     ;retorna su posición y
    ret                             ;sy valor ASCII en el
                                    ;registro AX.
TOMAR_TECLA ENDP
```

```
;
;                              INTRO_ESC
;  Espera  por el ingreso de  la tecla INTRO (CY=0) o de la
;  tecla ESC (CY=1).
;

INTRO_ESC    PROC
     call    tomar_tecla              ;Toma un caracter
     cmp     al,ENTER                 ;¿ INTRO ?
     clc                              ;Si, retorna con CY=0
     je      es1
     cmp     al,ESC                   ;¿ ESC ?
     stc                              ;Si, retorna con CY=1
     jne     intro_esc                ;No, entonces toma otro
es1:                                  ;caracter.
     ret
INTRO_ESC    ENDP

;
;                                SIONO
;  Espera por el ingreso de la tecla S (CY=0) o de la tecla
;  N o ESC (CY=1).
;

SIONO        PROC
     call    tomar_tecla              ;Toma un caracter
     and     al,5fh                   ;Convierte a mayúscula
     cmp     al,'S'                   ;¿ "S" ?
     clc                              ;Si, retorna con CY=0
     je      sn1
     cmp     al,'N'                   ;¿ "N" ?
     stc                              ;Si, retorna con CY=1
     je      sn1
     cmp     al,ESC                   ;¿ ESC ?
     stc                              ;Si, retorna con CY=1
     jne     siono                    ;No, entonces toma otro
sn1:                                  ;caracter
     ret
SIONO        ENDP

;
;                                TECLA
;  Chequea si una tecla ha sido presionada desde el teclado
;

TECLA        PROC    NEAR
     mov     bx,0

vaciar_teclado:
     mov     ah,1
```

```
    int     16h                     ;¿ Existe algún caracter
                                    ;disponible ?
    jz      vacio                   ;No, área de teclado
                                    ;vacía.
    xor     ah,ah                   ;Si, entonces...
    int     16h                     ;lo toma,
    mov     bx,ax                   ;lo guarda en BX
    jmp     vaciar_teclado          ;Y va a chequear  si  el
vacio:                              ;área el teclado ya está
                                    ;vacía.
    mov     ax,bx                   ;Al salir, AX contiene el
                                    ;código de la última
    ret                             ;tecla presionada
TECLA       ENDP


;
;                       MOSTRAR_MENU
;  Muestra  un texto en la línea del menú de la ventana del
;  Editor.
;

MOSTRAR_MENU        PROC    NEAR

    mov     di,19*160               ;Posición de video  donde
wrm0:                               ;se colocará el mensaje
    mov     es,sgmnto_de_video
    assume  es:nothing
    mov     bh,menu_attr_1          ;Mono-9h, Color-1Eh
    cld
    mov     cx,80                   ;Hasta 80 caracteres
wrm1:                               ;(1 línea)
    cmp     byte ptr [si],0         ;¿ Fin de la cadena ?
    je      wrm3                    ;Si, borra el resto de la
                                    ;línea.
    lodsb                           ;Toma el siguiente
                                    ;caracter
    cmp     al,'$'                  ;¿ Cambio de atributos de
                                    ;color ?
    jne     wrm2                    ;No, continúe
    xor     bh,menu_attr_2          ;Mono-0Eh, Color-1
    lodsb
wrm2:
    mov     bl,al
    call    escribir_palabra        ;Muestra  el  siguiente
                                    ;caracter
    dec     cl                      ;Decrementa  valor   de
                                    ;línea
    jmp     wrm1                    ;Continúa en el lazo
wrm3:
    jcxz    wrm5                    ;Sale si se envió una
                                    ;línea completa
    mov     bl,' '                  ;De lo contrario,
wrm4:
```

```
    call    escribir_palabra        ;Coloca espacios en
                                    ;blanco al resto de la
    loop    wrm4                    ;línea
wrm5:
    push    cs
    pop     es
    ret

MOSTRAR_MENU        ENDP


;
;   FIJAR_CURSOR: Posiciona el cursor según valor en DX
;

FIJAR_CURSOR:
    push    si
    xor     bh,bh                   ;Número de página
    mov     ah,2                    ;Función para posicionar
                                    ;el cursor
    int     10h                     ;Posiciona el cursor
    pop     si
    call    activar_cursor
    ret


;
;          ACTIVAR_CURSOR: Activa el cursor
;

ACTIVAR_CURSOR:
    mov     ah,1                    ;Función del BIOS para
                                    ;activar el cursor
    mov     cx,modo_del_cursor      ;Forma del cursor
    int     10h                     ;Activa el cursor
    ret


;
;          OCULTAR_CURSOR: Oculta el cursor
;

OCULTAR_CURSOR:
    mov     cx,2000h                ;Longitud nula del cursor
    mov     ah,1                    ;Función del BIOS
    int     10h                     ;Oculta el cursor
    ret


;
;   DIRECTORIO: Permite manipular archivos de disco(s)
;
```

```
DIRECTORIO  PROC    NEAR

    or      indicadores,100000b     ;Anula la salida del
                                    ;reloj

; Guarda unidad de disco actual.

    cld
    mov     ah,19h                  ;Obtiene la unidad de
    int     21h                     ;disco actual
    mov     disco_actual,al         ;y la almacena
    lea     si,dir_telex            ;Posición del directorio
                                    ;del telex
    cmp     byte ptr ds:[si+1],':'  ;¿ Cambio de unidad de
                                    ;disco ?
    jne     guardar_dir             ;No, guarda el directorio
    mov     ah,al                   ;actual.
    lodsb                           ;Si, ver a que unidad se
    and     al,5Fh                  ;debe cambiar.
    sub     al,'A'
    cmp     al,ah                   ;¿ Estamos en la unidad
                                    ;deseada ?
    je      guardar_dir             ;Si, guardar el directorio
                                    ;actual
    mov     dl,al                   ;No, entonces DL debe
                                    ;contener unidad de disco
    mov     ah,0eh                  ;Función para cambio de
    int     21h                     ;unidad de disco

; Guarda el Directorio actual.

guardar_dir:
    lea     si,current_dir          ;Posición para el
                                    ;directorio actual
    call    obtener_dir             ;Toma y guarda el
                                    ;directorio actual

;  Si el directorio TELEX existe, se realiza un cambio al
;  mismo.

    lea     dx,dir_telex            ;Cadena \telex
    mov     ah,3bh                  ;Función para cambio de
                                    ;directorio
    int     21h                     ;Cambia de directorio
    cmp     ax,3                    ;¿ Existe el directorio
                                    ;TELEX ?
    jnz     mensaje                 ;Si, continúe
    call    parlante                ;No, suena alarma

; Actualiza las variables.

mensaje:
    lea     si,tabla_de_variab      ;Inicializa todas las
    lea     di,fin_de_pagina        ;variables usadas por
    mov     cx,13                   ;DIRECTORIO
```

```
    cld
    rep     movsb

; Muestra el mensaje: 'Leyendo y ordenando directorio...'

    lea     si,leyendo              ;Mensaje.
    mov     dx,172ch                ;Posición  donde  se
                                    ;muestra el mensaje
    call    mostrar_texto           ;Mensaje

; Limpia el área del directorio.

    lea     di,dir_trabajo          ;Borra todo el espacio de
    mov     ax,2020h                ;memoria usado para el
    cld                             ;almacenamiento  de  los
lazo:                               ;nombres de los archivos
    stosw                           ;y el área para  ver  el
    cmp     di,offset dta           ;contenido de los
    jb      lazo                    ;archivos

;  Lee  los  nombres de los archivos  del  directorio  y  los
;  almacena como registros en el espacio reservado.

    call    leer_dir                ;Lee el  primer  archivo
                                    ;del directorio
    jnc     almacenar_nombre
    jmp     salir_dir               ;Retorna si el directorio
                                    ;está vacío.
almacenar_nombre:
    lea     di,nombres_de_archivos  ;Puntero de los  archivos
                                    ;del directorio
    lea     bp,fin_area_dir         ;Puntero  de  fin   de
                                    ;archivos.
    call    preparar_nombre         ;Almacena el nombre  del
                                    ;archivo tomado.
encontrar_sgte:
    mov     ah,4fh                  ;Continúa la busqueda  de
    int     21h                     ;archivos.
    jc      almacenar_contador      ;Si CY=1, no  hay  más
                                    ;archivos.
    inc     numero_de_archvs,       ;Incrementa contador  de
    call    preparar_nombre         ;archivos.
    cmp     di,bp                   ;Existe espacio para otro
                                    ;nombre de archivo ?
    jb      encontrar_sgte          ;Si, tome  el  siguiente
                                    ;nombre de archivo

; Demasiados archivos.

                                    ;No, entonces
    lea     si,demasiados           ;Muestra en la  posición
    mov     dx,182ch                ;dada por DX el mensaje:
    call    alerta                  ;"Existen demasiados
                                    ;archivos..."
                                    ;y solo procesa los
                                    ;archivos leidos.
```

```
; Almacena direcciones finales y ordena los nombres de los
; archivos.

almacenar_contador:
    mov     fin_offset,di           ;Almacena fin de nombres
    add     di,40                   ;de archivos.
    mov     paginas,di              ;También la dirección
                                    ;para la páginas
    mov     bx,numero_de_archvs     ;Toma contador de
                                    ;archivos
    cmp     bx,18                   ;¿Suficiente para llenar
                                    ;una página ?
    jae     listo                   ;Si, continúe
    add     paginas,800             ;No, ajusta página
    mov     ax,160                  ;Calcula último registro
    mul     bl
    add     ax,485
    mov     fin_de_pagina,ax

; Ahora estamos listos para mostrar el Directorio y el Menú.

listo:
    call    borrar_msg              ;Borra mensajes anteriores
    lea     di,dir_trabajo          ;Especificación
                                    ;[unidad]:[directorio]
    mov     ah,19h                  ;Unidad de disco activa
    int     21h
    add     al,'A'                  ;Convierte a valor ASCII
    stosb
    mov     al,':'                  ;Agrega ":"
    stosb
    mov     si,di
    call    obtener_dir             ;Toma el directorio
                                    ;activo
    call    borrar_ventana_dir      ;Limpia la memoria de
                                    ;video para formar la
                                    ;ventana del DIRECTORIO
    call    ventana_de_menu         ;Muestra la ventana del
                                    ;MENU.
obtener_tecla:

    call    actualizar_dir          ;Muestra la ventana del
                                    ;DIRECTORIO
    call    tomar_tecla             ;Toma un caracter desde
                                    ;el teclado
    mov     bx,ax                   ;Guarda caracter tomado
    cmp     ah,1                    ;¿ESC o tecla inferior ?
    jbe     funcion                 ;Si, chequear función
    cmp     ah,36h                  ;¿ Tecla de avance del
                                    ;cursor o superior ?
    jae     funcion                 ;Si, chequear función
    cmp     ah,1ch                  ;¿ Tecla INTRO ?
    jnz     obtener_tecla           ;No, ignore caracter
funcion:
    lea     di,tabla_dir
```

```
    mov     al,bh
    mov     cx,19                   ;19 comandos válidos
    repnz   scasb
    jnz     obtener_tecla           ;Si no es ninguno, ignora
    lea     di,subroutinas_dir+36   ;el caracter
    shl     cx,1
    sub     di,cx
    call    ds:[di]                 ;O si no, ejecuta la
                                    ;subroutina.
    jmp     short obtener_tecla     ;Actualiza la pantalla y
                                    ;toma el siguiente
                                    ;comando

;  Al finalizar la sesión del DIRECTORIO, se recuperan la
;  unidad de disco y el directorio que estaban activos al
;  inicio de la subroutina.

salirx:

    pop     ax

salir_dir:

; Unidad de disco actual.

    mov     dl,disco_actual         ;Recupera unidad de disco
                                    ;actual
    mov     ah,0eh                  ;Función parà cambiar de
    int     21h                     ;unidad de disco

; Directorio actual.

    lea     dx,current_dir          ;Recupera el directorio
                                    ;activo.
    mov     ah,3bh                  ;Función para cambio de
    int     21h                     ;directorio.
    call    abrir_ventana_editor    ;Recupera la ventana del
                                    ;EDITOR
    and     indicadores,11011111b   ;Activa la salida del
                                    ;Reloj.
    ret

DIRECTORIO  ENDP

;
;   OBTENER_DIR: Obtiene el nombre del Directorio actual
;

OBTENER_DIR:
    mov     byte ptr [si],'\'       ;El nombre del directorio
    inc     si                      ;debe comenzar con "\"
    xor     dl,dl                   ;Directorio actual
    mov     ah,47h                  ;Función para obtener
```

```
    int     21h                     ;el directorio actual.
    ret

;
;                          LEER_DIR
;  Comienza la búsqueda de archivos de un directorio
;  especifico. A la salida CY=0 indica que por lo menos
;  existe un archivo en ese directorio.
;

LEER_DIR    PROC    NEAR
    lea     dx,global               ;*.*
    mov     cx,7                    ;Atributo de archivos
                                    ;normales
    mov     ah,4eh                  ;Comienza la búsqueda de
    int     21h                     ;archivos.
    jc      no_archivos             ;¿ Directorio vacio ?
    ret                             ;No, sale con CY=0
no_archivos:                        ;Si, entonces
    lea     si,dir_vacio            ;Muestra el mensaje:
    mov     dx,172ch                ;"No se encontraron
    call    alerta                  ;archivos..."
    stc                             ;Sale con CY=1
    ret
LEER_DIR    ENDP

;
;  Las siguientes subroutinas controlan la barra y paginado
;  del listado del directorio.
;

SUBIR_BARRA:

    mov     bp,-160                 ;Sube la barra
    jmp     short fijar_barra

BAJAR_BARRA:
    mov     bp,160                  ;Baja la barra
fijar_barra:
    call    desplazar_barra
    ret

PG_UPX:
    mov     bp,-40*18               ;Sube la barra 21 lineas
    call    desplazar
    jmp     short top_bar

PG_DNX:
    mov     bp,40*18                ;Baja la barra 21 lineas

MOVE_PAGE:
```

```
        call    desplazar
        jmp     short bottom_bar

; Mueve la barra al inicio del listado del directorio.

HOME_BAR:
        mov     cur_offset,offset nombres_de_archivos

TOP_BAR:
        mov     si,485
        call    mover_barra
        ret

; Mueve la barra al final del listado del directorio.

FIN_BAR:
        mov     bx,fin_offset
        sub     bx,18*40
        cmp     bx,offset nombres_de_archivos
        jbe     bottom_bar
        mov     cur_offset,bx

BOTTOM_BAR:
        mov     si,fin_de_pagina
        sub     si,160
        call    mover_barra
        ret


;
;   DESPLAZAR: Esta subrotuina sube el contenido de video
;


DESPLAZAR:
        mov     si,cur_offset
        add     si,bp

ck_inferior:

        cmp     si,offset nombres_de_archivos
        jae     limite_superior

limite_inferior:

        mov     cur_offset,offset nombres_de_archivos
        jmp     short salir_desplazar

limite_superior:

        mov     bx,fin_offset
        cmp     bx,offset nombres_de_archivos+18*40
        ja      ck_superior
        mov     cur_offset,offset nombres_de_archivos
        jmp     short salir_desplazar
```

```
ck_superior:

    sub     bx,18*40
    cmp     si,bx
    jbe     fin_desplazar
    mov     si,bx

fin_desplazar:

    mov     cur_offset,si

salir_desplazar:

    ret

;
;  Esta subroutina mueve la barra entre el inicio y el final
;  de la página.
;

DESPLAZAR_BARRA:
    mov     si,linea_actual         ;Toma linea actual
    add     si,bp                   ;Agrega linea requerida
    mov     bp,-40                  ;Asume que est fuera  del
                                    ;inicio de la página.
    cmp     si,485                  ;¿ Es asi ?
    jb      desplazar_pagina        ;Si, desplaza la página
    mov     bp,40                   ;Realiza lo mismo para el
    cmp     si,fin_de_pagina        ;caso de final de página
    jae     desplazar_pagina
    call    mover_barra             ;Caso contrario, mueve la
    ret                             ;barra.

desplazar_pagina:
    call    desplazar
    ret

;
;     Esta subroutina realiza el movimiento de la barra
;

MOVER_BARRA:
    mov     bl,normal               ;Mono-07, Color-70h
    call    bar                     ;Anula barra actual
    mov     linea_actual,si
    mov     bl,invertido            ;Mono-70h, Color-1eh
    call    bar                     ;Fija nueva posición  de
    ret                             ;la barra.

BAR:
    mov     es,sgmnto_de_video
```

```
    mov     di,linea_actual         ;Posición de la barra
    mov     cx,39                   ;Longitud de la barra 39

sgte_bar:
    call    escribir_byte           ;Escribe atributo de  la
    loop    sgte_bar                ;barra.
    push    cs
    pop     es
    ret


;
;                       VER_ARCHIVO
; Esta  subroutina  muestra el contenido  de un archivo el
; mismo que puede ser observado línea a línea o por páginas
;

VER_ARCHIVO:
    call    obtener_nombre          ;Toma nombre del archivo
    lea dx,nombre_de_archivo_actual ;Abre el archivo
    mov     ax,3d00h
    int     21h
    mov     puntero_del_archivo,ax  ;Guarda el puntero  del
                                    ;archivo
    call    leer_primer_bloque
    mov     ultima_pagina,offset segundo_bloque
    mov     bp,paginas              ;Inicializa al puntero de
                                    ;página
    mov     puntero_de_linea,0      ;y al puntero de línea
    call    borrar_area_de_video    ;Limpia el área de video
    lea     si,view_menu            ;Menú de visualización de
                                    ;archivos
    mov     di,160*24               ;Posición del  menú  de
                                    ;visualización
    call    wrm0                    ;Muestra el menú

obtener_view:
    mov     si,ultima_pagina        ;Toma el tope de página
                                    ;actual
    mov     bh,puntero_de_linea     ;Toma línea actual
    cmp     bh,0                    ;¿ Línea cero ?
    jz      sgte_por_ver            ;Si, salte
    mov     indicador_video,0       ;No video

 page_offset:
    call    lineas
    dec     bh
    jnz     page_offset
sgte_por_ver:
    mov     bh,24                   ;Ahora muestra 25 líneas
    xor     di,di                   ;Comienza desde el  tope
                                    ;superior
    mov     indicador_video,1       ;Video
view_display:
```

```
        call    lineas
        dec     bh
        jnz     view_display

view_command:
        call    tomar_tecla             ;Toma un caracter  desde
                                        ;el teclado
        cmp     ah,1                    ;¿ ESC ?
        jz      view_fin                ;Si, salir
        cmp     ah,1ch                  ;¿ INTRO ?
        jz      view_fin                ;Si, salir
        cmp     ah,3bh                  ;¿ [F1] ?
        jnz     ck_editar               ;No, chequear opción para
                                        ;editar un archivo.
view_fin:
        mov     bx,puntero_del_archivo  ;Si, entonces debe cerrar
                                        ;el archivo
        mov     ah,3eh                  ;Función para cerrar  un
                                        ;archivo
        int     21h                     ;Cierra el archivo
        call    borrar_ventana_dir      ;Ahora  debe  volver  a
        call    ventana_de_menu         ;mostrat las ventanas del
                                        ;directorio y del menú.
        ret                             ;Finalmente, retorna

ck_editar:

        cmp     ah,3Ch                  ;¿ Editar texto ? [F2]
        jnz     ck_renombrar            ;No,
        call    view_request            ;Si, recupere ventanas de
                                        ;DIR y MENU
        jmp     editar_archivo          ;Ejecute subroutina  para
                                        ;editar archivo
ck_renombrar:
        cmp     ah,3dh                  ;¿ Renombrar  archivo  ?
                                        ;[F3]
        jnz     ck_mover                ;No,
        call    view_request            ;Si, recupere ventanas de
                                        ;DIR y MENU
        jmp     renombrar               ;Ejecute subroutina  para
                                        ;renombrar archivo.
ck_mover:
        cmp     ah,3eh                  ;¿ Mover archivo ? [F4]
        jnz     ck_borrar               ;No,
        call    view_request            ;Si, recupere ventanas de
                                        ;DIR y MENU
        jmp     mover_archivo           ;Ejecute subroutina  para
                                        ;mover archivo
ck_borrar:
        cmp     ah,3Fh                  ;¿ Borrar archivo ? [F5]
        jnz     ck_cambio_de_dir        ;No,
        call    view_request            ;Si, recupere ventanas de
                                        ;DIR y MENU
        jmp     borrar_archivo          ;Ejecute subroutina  para
                                        ;borrar archivo
```

```
ck_cambio_de_dir:
    cmp     ah,41h                  ;¿ Cambiar de directorio?
                                    ;[F7]
    jnz     cursor_view             ;No, chequear teclas de
                                    ;movimiento del cursor.
    call    view_request            ;Si, recupere ventanas de
                                    ;DIR y MENU
    jmp     cambiar_de_directorio   ;Y ejecute subroutina
                                    ;de cambio de directorio
cursor_view:
    lea     di,tabla_dir+11         ;Códigos de teclas de
                                    ;movimiento
    mov     al,ah                   ;AH contiene el código de
                                    ;la posición
    mov     cx,8                    ;8 opciones
    repnz   scasb                   ;Chequea opciones
    jnz     view_command            ;Si no es ninguna, toma
                                    ;otra tecla
    lea     di,view_table+14        ;Caso contrario,
    shl     cx,1
    sub     di,cx                   ;Toma la dirección de la
                                    ;subroutina de opción,
    call    ds:[di]                 ;Y la ejecuta
    jc      view_command            ;Salida de la subroutina
    jmp     obtener_view            ;con ESC.
view_request:
    call    view_fin                ;Muestra las ventanas DIR
    .                               ;y MENU.
    call    actualizar_dir          ;Actualiza el directorio
    ret

;
;   Las siguientes subroutinas realizan el control de la
;   visualización del archivo seleccionado.
;

; Subir una línea.

UP_LINE:
    cmp     puntero_de_linea,0      ;¿ Primera línea ?
    jnz     dec_row                 ;No, decremente una línea
    cmp     bp,paginas              ;¿ Primera página ?
    stc
    jz      up_line_fin             ;Si es asi, nada cambia
    call    up_pg                   ;O si no, sube una página
    mov     puntero_de_linea,24     ;y mueve línea a 24
dec_row:

    dec     puntero_de_linea        ;Disminuye línea
    clc                             ;Señal para actualizar la
up_line_fin:                        ;pantalla.
    ret
```

```
; Bajar una linea.

DN_LINE:
    call    ck_fin_de_archivo       ;¿ Final del archivo ?
    jnc     ck_row                  ;No, chequear linea
    ret                             ;Si, salir

ck_row:
    cmp     puntero_de_linea,23     ;¿ Ultima línea ?
    jb      inc_row                 ;No, entonces incremente
                                    ;línea
    call    dn_pg                   ;Si, bajar página
    mov     puntero_de_linea,-1     ;Línea al tope
inc_row:
    inc     puntero_de_linea
    clc
dn_line_fin:
    ret

; Subir una página.

UP_PG:
    cmp     bp,paginas              ;¿ Primera página ?
    jnz     dec_pg                  ;No, disminuya página
    cmp     puntero_de_linea,0      ;¿ Primera línea ?
    stc
    jz      up_pg_fin               ;Si es asi, salir
    mov     puntero_de_linea,0      ;0 si no mover a la
    jmp     short actualizar_pagina ;primera línea
dec_pg:
    mov     si,ultima_pagina        ;Toma página actual
    sub     si,ds:[bp]              ;Resta diferencia a la
                                    ;página previa
    cmp si,offset area_ver_archivos ;¿ Mas allá del tope
                                    ;del área ?
    jae     up_pg_ret               ;No, sube una página
    call    hacia_atras             ;Si, lee previos 4 KB
    jmp     short dec_pg            ;Trata de nuevo
up_pg_ret:
    dec     bp
    dec     bp                      ;Decrementa el puntero de
                                    ;página.
    mov     ultima_pagina,si        ;Almacena nueva posición
                                    ;de inicio.
actualizar_pagina:
    clc
up_pg_fin:
    ret

; Bajar una página.

DN_PG:
    mov     si,ultima_pagina        ;Página actual
    mov     indicador_video,0       ;No video
    mov     bh,24                   ;Sube 24 lineas
```

```
sgte_page:
    call    lineas
    dec     bh
    jnz     sgte_page
    call    ck_fin_de_archivo       ;¿ Final del archivo ?
    jc      noactualizar_pg         ;Si, salir
inc_pg:
    mov     dx,ultima_pagina        ;Caso contrario, guarda
                                    ;offset actual
    mov     ultima_pagina,si        ;Almacena nuevo offset
    mov     ax,si
    sub     ax,dx                   ;Calcula la diferencia
                                    ;entre páginas
    inc     bp                      ;Incrementa valor de p
    inc     bp                      ;página.
    mov     ds:[bp],ax              ;y almacena
    clc
dn_pg_fin:
    ret
noactualizar_pg:
    stc
    ret

; Inicio del archivo.

HOME_FILE:
    call    up_pg                   ;Sube de páginas hasta
    jnc     home_file               ;alcanzar el inicio del
    clc                             ;archivo.
    ret

; Fin del archivo.

FIN_FILE:
    call    dn_pg                   ;Baja de páginas hasta
    jnc     fin_file                ;alcanzar el final del
    clc                             ;archivo.
    ret


;
;                    CK_FIN_DE_ARCHIVO
;  Esta subroutina chequea si se alcanzó el final del
;  archivo.
;

CK_FIN_DE_ARCHIVO:
    cmp     si,fin_de_archivo
    jb      no_fin_de_archivo
cmp fin_de_archivo,offset area_ver_archivos+(cuatro_Kbytes*2)
    jae     no_fin_de_archivo
    stc
    ret
no_fin_de_archivo:
```

```
    clc
    ret


;
;                    EDITAR_ARCHIVO
;          Permite que un archivo pueda ser editado
;

EDITAR_ARCHIVO:
    call    obtener_nombre          ;Toma el nombre del
                                    ;archivo a editar
    jnc     editarlo                ;Si es posible, trata de
                                    ;editarlo
    call    parlante                ;De lo contrario, suena
                                    ;una alarma,
fin_editar:
    call    borrar_msg              ;Borra mensaje
    call    ocultar_cursor          ;Oculta el cursor
    ret                             ;y sale

editarlo:
    mov     dx,162Dh                ;Posición para mostrar un
                                    ;mensaje
lea si,nombre_de_archivo_actual     ;Nombre del archivo a
                                    ;editar
    call    mostrar_texto           ;Muestra el nombre del
                                    ;archivo
    lea     si, sera_editado        ;Mensaje:
    call    obtener_texto           ;'será editado..
                                    ;¿ Está seguro (S/N) ?'
    mov     dx,172Dh
    call    mostrar_texto           ;Muestra el mensaje
    call    activar_cursor          ;y activa el cursor

opcion:
    call    siono                   ;¿ Editar el archivo ?
    jc      fin_editar              ;No, salir
                                    ;Si, entonces toma el
lea dx,nombre_de_archivo_actual     ;nombre del archivo a
                                    ;editar
    mov     ax,3D00h                ;Función del DOS para
    int     21h                     ;abrir el archivo
    mov     bx,ax                   ;Puntero del archivo
    mov     cx,7800                 ;Leer 7800 bytes del
    lea     dx,nombres_de_archivos  ;archivo.
    mov     ah,3Fh                  ;Función para leer el
    int     21h                     ;archivo.
    mov     fin_de_archivo,ax       ;Guarda el puntero del
                                    ;archivo
    call    formar_texto            ;Prepara el texto
    mov     bx,puntero_del_archivo  ;Recupera puntero del
                                    ;archivo
    mov     ah,3Eh                  ;Función para cerrar el
```

```
    int     21h                         ;archivo.
    call    fin_editar                  ;Oculta el cursor
    lea     si,dir_trabajo              ;Directorio del archivo
    call    guardar_nuevo_nombre
    cmp     byte ptr es:[di-1],'\'
    je      no_bak
    mov     al,'\'
    stosb
no_bak:     mov     cx,12
    lea si,nombre_de_archivo_actual ;Nombre del archivo
    call    nombre_de_texto
    jmp     salirx


;
;                          LINEAS
;  Esta subroutina  forma el texto a líneas. Una  línea  es
;  marcada por un CR o cuando se alcanzó la última columna.
;


LINEAS:
    mov     cx,80                       ;80 columnas

lineas_sgtes:
    cmp     si,fin_de_archivo           ;¿ Fin del  archivo  ?,
                                        ;complete con espacios en
                                        ;blanco.
    jb      tomar_lineas
cmp fin_de_archivo,offset area_ver_archivos+(cuatro_Kbytes*2)
    jb      llenar_espacios
    push    bx                          ;Si se alcanzó el  final
    push    cx                          ;área de visualización,
    push    di                          ;guarda los punteros
    call    hacia_adelante              ;Lee los siguientes 2 KB
    pop     di                          ;y recupera los punteros
    pop     cx
    pop     bx

tomar_lineas:
    lodsb                               ;Toma un byte
    cmp     al,13                       ;¿ Código de CR ?
    jz      llenar_espacios             ;Si, completar la  línea
                                        ;con espacios en blanco.
    cmp     al,9                        ;¿Es un caracter de TAB ?
    jz      tabx                        ;Si, tabule
    cmp     al,10                       ;¿Es un avance de línea ?
    jz      lineas_sgtes                ;Si, salte a la siguiente
                                        ;línea
    push    cx                          ;Guarda el contador
    mov     cx,1
    call    ck_display                  ;muestra un caracter
    pop     cx
    loop    lineas_sgtes                ;y regresa a tomar  otro
    cmp     byte ptr [si],13            ;caracter.
```

```
    jnz     fin_lineas
    inc     si
fin_lineas:
    ret


TABX:
    push    cx                      ;Guarda contador
    dec     cx
    and     cx,7
    inc     cx                      ;Ajusta
    push    cx
    call    llenar_espacios         ;Siguiente posición de
    pop     ax                      ;TAB.
    pop     cx
    sub     cx,ax                   ;Si no está en la columna
                                    ;80 toma el siguiente
    jnz     lineas_sgtes            ;caracter.
    ret

LLENAR_ESPACIOS:

    mov     al,32                   ;Caracter de espacio en
ck_display:                         ;blanco.
    cmp     indicador_video,1       ;¿ Estamos escribiendo en
                                    ;la pantalla ?
    jnz     ck_fin_disp             ;No, salir
    mov     bl,al
    push    bx
    mov     es,sgmnto_de_video
byte_a_video:
    call    escribir_byte           ;Si, escribe CX espacios
    loop    byte_a_video            ;en blanco.
    pop     bx
    push    cs                      ;Deja ES apuntando al
    pop     es                      ;segmento del programa
ck_fin_disp:
    ret                             ;y sale.


;
;   Las siguientes subroutinas controlan el paginado de la
;   visualización del contenido de un archivo.
;


HACIA_ADELANTE:

    sub ultima_pagina,cuatro_Kbytes ;Ajusta página actual
    xor     cx,cx                   ;Mueve puntero de archivo
    mov     dx,cuatro_Kbytes        ;4 Kbytes hacia adelante
    call    mover_puntero

LEER_PRIMER_BLOQUE;
    lea     si, segundo_bloque
```

```
    lea     di,area_ver_archivos    ;Mueve la segunda mitad
                                    ;del área reservada para
                                    ;visualizar un archivo
    call    mover_bloque            ;a la primera mitad y lee
                                    ;2 KB.
    mov     cx,-1                   ;Retrocede el puntero del
    neg     dx                      ;archivo.
    call    mover_puntero
    ret

HACIA_ATRAS:

    add ultima_pagina,cuatro_Kbytes ;Ajusta página actual
    mov     cx,-1                   ;Retrocede el puntero del
    mov    dx,- (cuatro_Kbytes * 2) ;archivo 8 Kb.
    call    mover_puntero
    lea     si, area_ver_archivos   ;Mueve la primera  mitad
    lea     di,segundo_bloque       ;del área hacia la segunda
    call    mover_bloque            ;mitad y lee 2 KB.
    ret

MOVER_PUNTERO:

    mov     bx,puntero_del_archivo  ;Puntero del archivo.
    mov     ax,4201h                ;Función del  DOS  para
    int     21h                     ;mover el puntero del
    ret                             ;archivo.

MOVER_BLOQUE:

    mov     dx,si                   ;Guarda puntero
    mov     cx,cuatro_Kbytes / 2    ;Mueve 2 Kb.
    rep     movsw
    mov     bx,puntero_del_archivo
    mov     cx,cuatro_Kbytes        ;Lee 4 Kb.
    mov     ah,3Fh
    int     21h
    mov     si,dx
    mov     dx,ax
    add     ax,offset segundo_bloque
    mov     fin_de_archivo,ax       ;Fin del rea de
    ret                             ;visualización.
```

```
;
;   BORRAR_ARCHIVO
;   Borra un archivo del directorio activado
;
```

```
BORRAR_ARCHIVO:

    call    obtener_nombre          ;Prepara  nombre   del
                                    ;archivo
    jc      no_permitido            ;Error si no es permitido
```

```
    cmp     senal_verific,0         ;¿ Verificar borrado del
                                    ;archivo ?
    jz      borrelo                 ;No, entonces borrar
                                    ;archivo
    call    borrar_msg
    mov     dx,162dh                ;Si,
    lea     si, nombre_de_archivo_actual
    call    mostrar_texto           ;Muestra el mensaje de
                                    ;advertencia:
    lea     si, sera_borrado
    call    obtener_texto
    mov     dx,172dh
    call    mostrar_texto           ;[Archivo] será borrado.
                                    ;¿ Está seguro S/N ?
    call    activar_cursor
    call    siono                   ;¿ Borrar el archivo ?
    jc      fin_borrar_archivo      ;No, salir
borrelo:                            ;Si, entonces obtiene el
    lea dx,nombre_de_archivo_actual ;nombre del archivo a
                                    ;borrar
    mov     ah,41h                  ;Función para borrar
                                    ;archivo
    int     21h                     ;Borra el archivo
    jc      no_permitido
    call    sacar_del_listado       ;Saca del listado el
fin_borrar_archivo:                 ;nombre del archivo
    call    ocultar_cursor          ;Oculta el cursor
    call    borrar_msg              ;Borra el mensaje de
    ret                             ;advertencia
no_permitido:
    call    parlante                ;Alarma cuando exista
                                    ;algún error
    jmp     short fin_borrar_archivo

;
;   CAMBIAR_DE_DIRECTORIO
;   Permite ejecutar un cambio de directorio.
;

CAMBIAR_DE_DIRECTORIO:
    pop     ax
    call    borrar_msg              ;Borra     mensajes
    mov     dx,162ch                ;anteriores.
    lea     si,cambio_de_dir        ;Muestra el mensaje:
    call    mostrar_texto           ;'Cambio de directorio:'
    call    activar_cursor          ;Activa el cursor
    call    borrar_entrada_anterior ;Prepara el área de
                                    ;alma enamiento para la
                                    ;entrada del  nuevo
key_dir:                            ;directorio.
    call    tomar_tecla             ;Toma un caracter  desde
                                    ;el teclado.
    cmp     al,ESC                  ;¿ Salir ?
```

```
    jz      fin_dr
    cmp     al,'?'                  ;No,   ignora    los
    jz      key_dir                 ;caracteres * y ?
    cmp     al,'*'
    jz      key_dir
    cmp     al,ENTER                ;¿ Fin del  nombre  del
                                    ;directorio ?
    jz      dir_listo               ;Si, listo
    cmp     al,8                    ;¿ Corregir entrada ?
    jnz     no_bs1x                 ;No, evitar espacios  en
                                    ;blanco
    call    corregir                ;Si, entonces retroceder
    jmp     short  key_dir          ;y tomar siguiente entrada
no_bs1x:
    cmp     al,' '                  ;Ignora códigos menores
    jbe     key_dir                 ;a 20h
    cmp     di,offset sendero + 35  ;¿ Hay espacio para  otro
                                    ;caracter ?
    jz      key_dir                 ;No, esperar por INTRO  o
                                    ;corrección
    stosb                           ;Si, almacénelo.
    call    escribir_texto          ;Muestra el  caracter
                                    ;ingresado.
    jmp     short  key_dir          ;y salta a  tomar  más
                                    ;entradas.
dir_listo:
    cmp     di,offset sendero       ;¿ Se ingresó algún
    jnz     comparar                ;caracter ?
fin_dr:
    call    ocultar_cursor          ;No, entonces oculta  el
                                    ;cursor,
    call    borrar_msg              ;Borra  el  mensaje  de
                                    ;cambio de directorio,
    jmp     obtener_tecla           ;y sale.
comparar:
    mov     ah,19h                  ;Toma la unidad de  disco
    int     21h                     ;actual
    mov     disco,al                ;y la guarda
    mov     byte ptr ds:[di],0      ;Cadena ASCIIZ
    lea     si,sendero              ;Posición   de   nuevo
almacenar_SI:                       ;directorio.
    push    si

sgte_parametro:
    lodsb
    cmp     al,0
    je      fin_parametro
    cmp     al,':'                  ;Chequea si se ingresó un
    jnz     sgte_parametro          ;cambio de unidad disco
    mov     dl,byte ptr [si-2]      ;Si es así, toma caracter
    and     dl,5fh                  ;de la unidad de disco  y
    sub     dl,'A'                  ;lo convierte a valor
    mov     ah,0eh                  ;decimal. (A=0, B=1, etc)
    int     21h                     ;Cambia  de  unidad  de
    pop     ax                      ;disco
```

```
    jmp     short almacenar_SI
fin_parametro:
    pop     dx
    mov     bl,byte ptr ds:[si-2]
    cmp     bl,':'
    jz      verificar_dir
    mov     ah,3bh                  ;Función del  DOS  para
    int     21h                     ;cambio de directorio
    jnc     verificar_dir           ;Si no  ocurrió  error,
                                    ;verifica el contenido
                                    ;del directorio.
    lea     si,dir_erroneo          ;Caso contrario,  muestra
    mov     dx,182ch                ;el mensaje:
    call    alerta                  ;'Directorio inválido...'
    jmp     restablecer             ;y deja las  cosas  como
                                    ;estaban
verificar_dir:
    call    leer_dir                ;Lee el  contenido  del
    jnc     nuevo_dr                ;directorio.
restablecer:
    mov     dl,disco                ;Si ocurrió algún error,
    mov     ah,0eh                  ;o si el directorio está
    int     21h                     ;vacío, recupera unidad
    jmp     obtener_tecla           ;de disco y sale
nuevo_dr:
    call    ocultar_cursor          ;Oculta el cursor.
    jmp     mensaje                 ;y sale


;
;          RENOMBRAR: Renombra un archivo
;


RENOMBRAR:

    call    obtener_nombre          ;Prepara  nombre   del
                                    ;archivo
    jc      no_posible              ;Error si no es permitido
    call    borrar_msg
    mov     dx,162ch
    lea     si, nuevo_nombre        ;Muestra el mensaje:
    call    mostrar_texto
    lea si,nombre_de_archivo_actual ;"Nuevo  nombre   para
    call    obtener_texto           ;[nombre_del_archivo]
    call    activar_cursor
    call    borrar_entrada_anterior ;Borra la última entrada

;Acepta el ingreso de un nuevo nombre para el archivo

ck_tecla:
    call    tomar_tecla             ;Toma un caracter
    cmp     al,ESC                  ;¿ ESC ?
    je      fin_renombrar           ;Si, salir
    cmp     al,':'                  ;Ignora los caracteres  "
```

```
    je      ck_tecla                ;: \ * ? "
    cmp     al,' '
    je      ck_tecla
    cmp     al,'\'
    je      ck_tecla
    cmp     al,'?'
    je      ck_tecla
    cmp     al,'*'
    je      ck_tecla
    cmp     al,ENTER                ;¿ INTRO ?
    je      renombrelo              ;Si, renombrar archivo
    cmp     al,8                    ;¿ Corregir ?
    jne     no_bs1
    call    corregir                ;Si, retrocede y borra
    jmp     short  ck_tecla         ;Toma el  siguiente
no_bs1:                             ;caracter.
    cmp     al,32                   ;¿ ASCII 32 o mayor ?
    jb      ck_tecla                ;No, ignore caracter.
    cmp     di,offset sendero + 12  ;¿ Fin del campo de
                                    ;entrada ?
    jz      ck_tecla                ;Si, esperar por INTRO
    stosb                           ;No, entonces almacena
                                    ;caracter,
    call    escribir_texto          ;y lo muestra en la
                                    ;pantalla
    jmp     short  ck_tecla         ;toma el  siguiente
                                    ;caracter
renombrelo:
    call    renombrar_archivo       ;Renombra el archivo
    jc      no_posible              ;Alarma si el intento es
                                    ;ilegal
    lea     dx,sendero              ;De lo contrario, coloca
                                    ;el nuevo nombre en el
                                    ;listado
    mov     cx,7                    ;Vuelve a leer el archivo
    mov     ah,4Eh                  ;renombrado para fijar la
    int     21h                     ;fecha y hora de creación
    mov     di,cur_file             ;del mismo.
    call    preparar_nombre
fin_renombrar:
    call    ocultar_cursor          ;Oculta el cursor
    call    borrar_msg              ;borra mensajes
    ret                             ;y sale
no_posible:
    call    parlante                ;Alarma cuando exista
    jmp     short fin_renombrar     ;algún error.


;
;   RENOMBRAR_ARCHIVO: Renombra un archivo
;

RENOMBRAR_ARCHIVO:
    mov     byte ptr ds:[di],0      ;Cadena ASCIIZ para nuevo
```

```
                                        ;nombre
    lea dx,nombre_de_archivo_actual     ;DX al nombre del archivo
    lea     di,sendero                  ;DI al nuevo nombre  para
                                        ;el archivo
    mov     ah,56h                      ;Función del  DOS  para
                                        ;renombrar un archivo
    int     21h                         ;Renombra el archivo.
    ret

;
;   ┌──────────────────────────────────────────────┐
;   │                MOVER_ARCHIVO                 │
;   │     Mueve un archivo desde un directorio a otro.  │
;   └──────────────────────────────────────────────┘

MOVER_ARCHIVO:
    call    obtener_nombre              ;Prepara el  nombre  del
                                        ;archivo
    jc      no_valido                   ;Si el archivo es oculto,
                                        ;sale
    call    borrar_msg                  ;Borra mensajes
    mov     dx,162Ch
    lea     si, nuevo_dir               ;Muestra el mensaje:
    call    mostrar_texto               ;'Nuevo directorio para '
    lea si,nombre_de_archivo_actual     ;Nombre del archivo
    call    obtener_texto
    call    activar_cursor              ;Activa el cursor
    call    borrar_entrada_anterior     ;Prepara el área ingreso
otro_dir:                               ;de un nuevo directorio.
    call    tomar_tecla                 ;Toma un caracter
    cmp     al,ESC                      ;¿ Salir ?
    jz      fin_mover
    cmp     al,':'                      ;No, ignora  cambio  de
    jz      otro_dir                    ;unidad de disco.
    cmp     al,ENTER                    ;¿ Listo ?
    jz      moverlo                     ;Si, mover el archivo.
    cmp     al,8                        ;¿ Corregir entrada
    jnz     no_bs2                      ;anterior ?
    call    corregir                    ;Si, retroceder
    jmp     short otro_dir              ;y tomar otro caracter
no_bs2:
    cmp     al,32                       ;¿ ASCII inferior a 20h?
    jb      otro_dir                    ;Si, ignórelo
    cmp     di,offset sendero + 35      ;¿ Hay espacio para  otro
                                        ;caracter ?
    jz      otro_dir                    ;No, esperar por  ingreso
                                        ;de INTRO o corrección.
    stosb                               ;Si, entonces lo almacena
    call    escribir_texto              ;Lo  muestra   en   la
    jmp     short  otro_dir             ;pantalla y salta a tomar
                                        ;otro caracter
moverlo:
    lea     si, sendero                 ;Cadena ingresada
    cmp     byte  ptr  ds:[si],32       ;¿ Se ingresó algún
```

```
                                        ;caracter ?
    jz      no_valido                   ;No, salir

    lea si,nombre_de_archivo_actual ;Apunta al  nombre  del
    mov     al,'\'                      ;archivo.
    cmp     ds:[di-1],al
    jz      agregar_nombre              ;Si es necesario,
    stosb                               ;agrega un '\' al  final
agregar_nombre:                         ;del nombre de la
                                        ;cadena ingresada
    movsb                               ;Une  cadena-nombre  del
    cmp     byte ptr ds:[si],0          ;archivo.
    jnz     agregar_nombre
    call    renombrar_archivo           ;Renombra el archivo
    jc      no_valido                   ;Sale si  ocurrió  algún
                                        ;error
    call    sacar_del_listado           ;Caso contrario, borra
fin_mover:                              ;del listado el nombre
                                        ;del archivo.
    call    ocultar_cursor              ;Oculta el cursor
    call    borrar_msg                  ;Borra los mensajes
    ret                                 ;y sale

no_valido:
    call    parlante                    ;Alarma si existió alguna
    jmp     short fin_mover             ;acción ilegal.
```

```
;
;                          CONFIRMAR
;  Esta  subroutina  conmuta  el mensaje de confirmación de
;  borrado de los archivos.
;
```

```
CONFIRMAR:

    xor     senal_verific,1             ;Conmuta indicador
    lea     di,menu_dir+(9*20)+32       ;Posicion de  SI/NO
    lea     si, on                      ;Asume 'SI'
    cmp     senal_verific,1             ;¿ Está en 'SI' ?
    jz      conmute                     ;Si, continúe
    lea     si, off                     ;No, muestre un 'NO'

conmute:
    movsw
    call    ventana_de_menu             ;Muestra 'SI' o 'NO'
    ret
```

```
;
;          ORDENAR_POR_NOMBRE, ORDENAR_POR_FECHA
;  Posiciona  el  puntero de ordenamiento de los registros,
;  ya sea por nombre o por fecha.
;
```

```
ORDENAR_POR_NOMBRE:

    mov     orden_offset,0          ;Posiciona en el nombre
                                    ;del archivo
    jmp     short orden_msg         ;Mostrar mensaje de
                                    ;ordenamiento

ORDENAR_POR_FECHA:

    mov     orden_offset,29         ;Posiciona en la fecha de
                                    ;creación del archivo.
orden_msg:
    call    borrar_msg              ;Borra mensajes anteriores
    mov     dx,1731h                ;Posición para el mensaje
    lea     si,leyendo+10           ;Muestra el mensaje:
    call    mostrar_texto           ;"Ordenando directorio"

;
;                        ORDENAR
;  Esta subroutina ordena por nombre o por fecha el listado
;  de los archivos del directorio.
;

ORDENAR:

    cmp     numero_de_archvs,1      ;¿Existe un solo archivo ?
    jz      salir_de_orden          ;Si, entonces salir
    mov     dx,fin_offset           ;No, fija DX al final del
                                    ;listado.
    sub     dx,40                   ;Inicio del último reg.
sgte_paso:
    mov     indicador_orden,0       ;Indicador de orden
    lea     bx,nombres_de_archivos  ;Inicio del listado

sgte_orden:
    mov     si,bx                   ;Fuente y destino
    add     si,orden_offset         ;Posición del nombre o
                                    ;de la fecha de creación
    mov     di,si                   ;del archivo.
    add     di,40                   ;Registro siguiente
    cmp     orden_offset,29         ;¿ Ordenar por fecha ?
    jz      ordenar_fecha           ;Si, hacerlo
    mov     cx,12                   ;No,
compare:                            ;12 bytes del nombre del
                                    ;archivo.
    repz    cmpsb                   ;Compara los registros
    jbe     fin_orden               ;Comparar los siguientes
                                    ;registros si ya están
intercambie:                        ;ordenados
    mov     si,bx                   ;Caso contrario, recupera
    mov     di,bx                   ;los punteros iniciales
    add     di,40
    mov     cx,20                   ;Fija 40 bytes por
```

```
sgte_intercambio:                       ;intercambiar.
    mov     ax,[di]                     ;Toma 2 bytes del segundo
                                        ;registro
    movsw                               ;Mueve 2 bytes del primer
                                       ;registro hacia el segundo
                                        ;registro
    mov     [si-2],ax                   ;Y mueve los 2 bytes  de
                                        ;AX al primer registro.
    loop    sgte_intercambio            ;Continúa el intercambio
    mov     indicador_orden,1           ;Indicador de intercambio

fin_orden:
    add     bx,40                       ;Apunta al sig. registro
    cmp     bx,dx                       ;¿ Fin del listado ?
    jb      sgte_orden                  ;No, continuar ordenando
    sub     dx,40                       ;Si, disminuye puntero
                                        ;de fin del listado
    cmp     indicador_orden,1           ;¿ Ocurrió por lo menos
                                        ;un intercambio ?
    jz      sgte_paso                   ;Si, continuar ordenando

salir_de_orden:                         ;No, entonces se borra el
    call    borrar_msg                  ;mensaje de ordenamiento
    ret                                 ;y retorna

ORDENAR_FECHA:

    mov     cx,2                        ;Primero compara el año
    repz    cmpsb                       ;de  creación  de  los
                                        ;archivos
    ja      intercambie                 ;Si el primer puntero  es
                                        ;mayor, intercambia los
                                        ;registros
    jnz     fin_orden
    sub     si,8                        ;Caso contrario, fija los
    sub     di,8                        ;punteros en la posición
    mov     cx,5                        ;de mes y día.
    repz    cmpsb                       ;Los compara
    ja      intercambie                 ;Si el primer puntero  es
    jnz     fin_orden                   ;mayor, intercambia los
                                        ;registros;
    add     si,10                       ;De otro modo, fija  los
    add     di,10                       ;punteros en la posición
                                        ;del meridiano.
    cmpsb                               ;Compara 'am' o 'pm'
    ja      intercambie                 ;Si es necesario,
    jnz     fin_orden                   ;intercambia los
                                        ;registros
    sub     si,6                        ;Posición de la hora
    sub     di,6
    mov     cx,5
    cmp     word ptr [si],3231h         ;Es el caso especial 12:?
    jz      ck_meridiano                ;Si, ver si son iguales
    cmp     word ptr [di],3231h         ;¿ Es el destino '12:' ?
    jnz     compare                     ;No, compare normalmente
```

```
    jmp     intercambie             ;Si, intercambie los
ck_meridiano:                       ;registros.
    cmpsw                           ;Son ambos '12:'
    jnz     fin_orden               ;No, comparar los
                                    ;siguientes registros
    mov     cx,3                    ;Caso contrario, compare
    jmp     short compare           ;los minutos.

;
;                       OBTENER_NOMBRE
;  Esta subroutina obtiene el archivo seleccionado y lo
;  convierte al formato DOS.
;

OBTENER_NOMBRE:

    mov     si,cur_offset           ;Toma el tope de página,
    mov     ax,linea_actual         ;la localización de la
    sub     ax,485                  ;barra,
    mov     cl,2
    shr     ax,cl
    add     si,ax                   ;Y ajusta
    mov     cur_file,si             ;Guarda puntero
    push    si
    lea di,nombre_de_archivo_actual ;Almacena los 8 primeros
    mov     cx,8                    ;caracteres.
    call    almacenar_bytes
    inc     si
    cmp     byte ptr ds:[si],32     ;¿ Fin del nombre del
                                    ;archivo ?
    jz      fin_nombre              ;Si, chequear si el
                                    ;archivo es oculto
    mov     al,'.'                  ;No, entonces agrega un
    stosb                           ;'.'
    mov     cx,3                    ;Hasta 3 caracteres como
    call    almacenar_bytes         ;extensión del archivo
fin_nombre:
    mov     byte ptr ds:[di],0      ;Forma cadena ASCIIZ
    pop     si                      ;Recupera puntero
    cmp     byte ptr [si+39],'H'    ;¿ Es un archivo oculto ?
    stc
    jz      nombre_error            ;Si, salir con CY=1
    clc                             ;No, salir con CY=0
nombre_error:
    ret
almacenar_bytes:                    ;Toma un caracter
    lodsb
    cmp     al,32                   ;Si es un espacio en
    jz      no_almacene             ;blanco, lo ignora
    stosb                           ;Caso contrario, lo
no_almacene:                        ;almacena.
    loop    almacenar_bytes         ;Continúa CX veces.
    ret
```

```
;
;                        BORRAR_ENTRADA_ANTERIOR
;  Esta subroutina borra la última entrada del usuario y
;  deja listo el espacio para almacenamiento de un nuevo
;  nombre de un archivo o directorio ingresado por el
;  usuario.
;

BORRAR_ENTRADA_ANTERIOR:

    mov     dx,172Ch
    call    fijar_cursor          ;Fija y activa el cursor
    lea     di,sendero            ;Area de entrada
    mov     ax,2020h              ;espacios en blanco
    mov     cx,18                 ;36 bytes
    rep     stosw                 ;borra
    lea     di,sendero            ;Fija puntero al inicio
    ret                           ;del área y sale

;
;                           CORREGIR
;  Subroutina de retroceso. Usada para corrección de ingreso
;  de caracteres desde el teclado.
;

CORREGIR:

    cmp     di,offset sendero     ;¿ Estamos al inicio del
                                  ;campo ?
    jz      fin_corregir          ;Si, salir
    dec     di                    ;No, entonces disminuye
    lea     si,codigos_corregir   ;el puntero y borra el
    call    obtener_texto         ;último caracter ingresado
fin_corregir:
    ret

;
;                        SACAR_DEL_LISTADO
;  Esta subroutina borra el nombre de un archivo desde el
;  listado del directorio.
;

SACAR_DEL_LISTADO:
    mov     di,cur_file           ;Apunta al nombre del
    mov     si,di                 ;archivo y mueve todos
    add     si,40                 ;los registros siguientes
                                  ;una posición hacia
sgte_registro:                    ;arriba.
    mov     cx,20                 ;40 bytes de cada
                                  ;registro
```

```
    rep     movsw                   ;sube un registro
    cmp     di,fin_offset           ;¿ Fin del listado ?
    jb      sgte_registro           ;No, mover el siguiente
                                    registro
    sub     di,40                   ;Si, entonces actualiza
    mov     fin_offset,di           ;el nuevo puntero de fin
    xor     bp,bp                   ;de directorio.
    call    desplazar               ;Actualiza la pantalla
    dec     numero_de_archvs        ;Decrementa contador de
                                    ;archivos
    jnz     mas_archivos            ;Si el directorio queda
                                    ;vacío, sale
    ret
mas_archivos:
    cmp     numero_de_archvs,18     ;¿ Página completa ?
    jae     remove_fin              ;Si, salte
    sub     fin_de_pagina,160       ;O si no, ajusta final de
    mov     si,fin_de_pagina        ;página
    sub     si,160
    cmp     si,linea_actual         ;¿ Está la barra bajo el
                                    ;listado del directorio ?
    ja      remove_fin              ;No, salir
    call    mover_barra             ;Si, sube la barra una
                                    ;posición,
remove_fin:
    call    contar_archivos         ;y actualiza en la
    ret                             ;pantalla el contador de
                                    ;de archivos.
```

```
;
;                          CONTAR_ARCHIVOS
;  Esta  subroutina calcula y muestra el número de archivos
;  existentes en el directorio.
;
```

```
CONTAR_ARCHIVOS:

    lea     di,archivos             ;Blanquea contador previo
    mov     ax,2020h
    stosw
    mov     ax,numero_de_archvs     ;Toma número de archivos
    mov     bl,10                   ;Lo convierte a decimal
    std
sgte_conteo:
    div     bl
    xchg    al,ah
    add     al,'0'                  ;y luego a ASCII
    stosb
    xchg    al,ah
    xor     ah,ah
    cmp     ax,0
    jnz     sgte_conteo .
    cld
```

```
    mov     dx,160eh                ;Línea 22, columna 13
    lea     si, archivos            ;Muestra el contador de
    call    mostrar_texto           ;archivos.
    call    ocultar_cursor          ;oculta el cursor
    ret                             ;y sale

;
;                          VENTANA_DE_MENU
;  Muestra la ventana y el contenido del menú del DIRECTORIO
;

VENTANA_DE_MENU:

    mov     es,sgmnto_de_video      ;ES al segmento de video
    mov     bx,(3*160)+98           ;Posición inicial de la
                                    ;ventana
    mov     dx,1411h                ;DH = 20 columnas,
                                    ;DL = 17 líneas
    mov     al,at_brd_menu          ;Monoc-7, Color-11
    call    borde_ventana           ;Forma la ventana del
                                    ;menú.
    lea     si, menu_dir            ;Contenido de la ventana
    mov     di,(3*160)+100          ;Posición inicial para el
                                    ;contenido
    mov     puntero_de_linea,18     ;18 líneas del menú
    mov     bh,70h                  ;Atributo para la primera
                                    ;línea.
sgte_linea_menu:
    mov     cx,20                   ;20 caracteres por línea
    mov     atributo,8
    call    mostrar_linea           ;Muestra el contenido de
                                    ;la línea
    add     di,120                  ;Posición inicial para la
                                    ;siguiente línea.
    dec     puntero_de_linea        ;¿ Se han mostrado ya las
                                    ;17 líneas ?
    mov     bh,7                    ;Atributo para las líneas
                                    ;restantes
    jnz     sgte_linea_menu         ;No, continúe con la
                                    ;línea siguiente.
    mov     bl,invertido            ;Fondo invertido
    call    bar                     ;Muestra la barra del
                                    ;directorio
    push    cs                      ;Actualiza ES al segmento
    pop     es                      ;del programa.
    ret

;
;                          ACTUALIZAR_DIR
;  Muestra la ventana y listado de los archivos del
;  DIRECTORIO.
;
```

```
ACTUALIZAR_DIR:
    xor     bp,bp
    mov     es,sgmnto_de_video      ;ES al segmento del video
    xor     bx,bx                   ;Posición inicial de la
                                    ;ventana
    mov     dx,2917h                ;DH = 41 columnas,
                                    ;DL = 23 líneas.
    mov     al,at_brd_dir           ;Monoc-70h, Color-7Ah
    call    borde_ventana           ;Forma la ventana del
                                    ;directorio
    mov     di,3*160+4              ;Posición para mostrar
                                    ;nombres de los archivos.
    mov     puntero_de_linea,18     ;18 nombres de archivo.
    mov     si,cur_offset           ;Puntero inicial.
sgte_linea_dir:
    mov     cx,40                   ;40 caracteres por linea
sgte_caracter:
    lodsb                           ;Toma un caracter,
    mov     bl,al                   ;Lo deposita en BL
    call    escribir_byte           ;y lo muestra en la
                                    ;pantalla
    loop    sgte_caracter           ;repite la secuencia CX
                                    ;veces
    add     di,80                   ;Posición inicial de la
                                    ;siguiente linea.
    dec     puntero_de_linea        ;¿ Se han mostrado ya las
                                    ;18 líneas ?
    jnz     sgte_linea_dir          ;No, continúe con la
                                    ;línea siguiente
    push    cs                      ;Si, entonces fija ES
    pop     es                      ;al segmento del programa
    mov     dx,0102h                ;DH = linea 1,
                                    ;DL = columna 2.
    lea     si,directorio_de        ;Nombre del directorio
    call    mostrar_texto           ;Muestra el nombre del
    lea     si, dir_trabajo         ;directorio.
    call    obtener_texto
    call    contar_archivos         ;Muestra el número de
    ret                             ;archivos del directorio.

;
;                          VENTANA_TELEX
;     Ventana que muestra el envio/recepción de un caracter.
;

VENTANA_TELEX:
    mov     es,sgmnto_de_video      ;ES:DI hacia la memoria
    assume  es:nothing              ;de video.
    mov     bx,53*2                 ;Posición inicial de la
                                    ;ventana de TELEX.
    mov     dx,1901h                ;DH:25 Columnas
                                    ;DL:1 Linea
    mov     al,4Fh                  ;Mono-70h, Color-1Eh
```

```
    call    borde_ventana
    ret

;
;   VENTANA_CNFG: Ventana de Configuración para el TELEX
;

VENTANA_CNFG:

    mov     es,sgmnto_de_video      ;ES:DI hacia la memoria
    assume  es:nothing              ;de video.

    mov     bx,160*7+23*2           ;Posición inicial de la
                                    ;ventana de Configuración
    mov     dx,2109h                ;DH:33 Columnas
                                    ;DL:09 Lineas
    mov     al,11                   ;Atributo
    call    borde_ventana
    call    borrar_ventana_cnfg     ;Borra espacio usado por
    mov     di,160*7+24*2           ;la ventana.
    lea     si, menu_cnfg
    mov     bh,70h                  ;Atributo de la primera
    mov     atributo,8              ;línea.
    mov     cx,33                   ;33 caracteres por linea
    call    mostrar_linea           ;Muestra:

    ret                             ;'Configuración-PC/XT/AT'

;
;                       MOSTRAR_LINEA
;   Muestra una cadena de caracteres en la memoria  de video,
;   cambiando el atributo de color de alguno de ellos.
;

MOSTRAR_LINEA:

    lodsb
    cmp     al,'$'                  ;Si encuentra un '$',
    jne     lst1                    ;cambia el atributo de
    xor     bh,atributo             ;los sgtes caracteres a
                                    ;mostrar.
    lodsb                           ;Toma el caracter...

lst1:

    mov     bl,al
    call    escribir_palabra        ;y lo envia a la memoria
                                    ;de video
    loop    mostrar_linea           ;Este lazo se ejecuta CX

    ret                             ;veces.
```

```
;
;                          BORDE_VENTANA
;                    Crea el borde de una ventana
;
;    Entrada:          BX - Esq. sup. izq. de la ventana
;                      DH - Long. Horz.   DL - Long. Vert.
;                      AL - Atributo del borde
;
```

```
BORDE_VENTANA:

    push    bx
    mov     di,bx                   ;DI en posición inicial
    cld                             ;de la ventana a formar
    mov     bh,al                   ;Atributo del borde de la
                                    ;ventana.
    mov     bl,'╔'                  ;Esquina sup. izquierda
    call    escribir_palabra        ;Muestra esquina
    xor     ch,ch
    call    linea_horizontal        ;Forma la línea superior
    mov     bl,'╗'                  ;Esquina superior derecha
    call    escribir_palabra        ;Muestra esquina.
    call    linea_vertical          ;Luego forma la columna
    add     di,158                  ;de la derecha.
    mov     bl,'╝'                  ;Esquina inferior derecha
    call    escribir_palabra        ;Muestra esquina.
    pop     di                      ;Vuelve DI a la posición
    add     di,2                    ;inicial de la ventana
                                    ;+ 1 posición
    call    linea_vertical          ;y forma la columna de la
                                    ;izquierda.
    add     di,158                  ;Después,
    mov     bl,'╚'                  ;Escribe el caracter de
    call    escribir_palabra        ;la esquina inferior
                                    ;izquierda.
    call    linea_horizontal        ;Finalmente, forma la
    ret                             ;línea inferior.
```

```
;
;                          LINEA_HORIZONTAL
;                Forma una linea horizontal de una ventana
;
```

```
LINEA_HORIZONTAL:

    mov     bl,'═'                  ;Caracter del borde de la
                                    ;ventana
    mov     cl,dh                   ;Longitud de la linea
lh:                                 ;horizontal
    call    escribir_palabra        ;Escribe el caracter CX
    loop    lh                      ;veces.
    ret
```

```
;
;                          LINEA_VERTICAL
;                 Forma una línea vertical de una ventana
;

LINEA_VERTICAL:

    mov     bl,'||'                 ;Caracter del borde
    mov     cl,dl                   ;Longitud de la línea
vt_der:                             ;vertical.
    add     di,158                  ;Siguiente posición
    call    escribir_palabra        ;Escribe el caracter CX
    loop    vt_der                  ;veces.

    ret

;
;                          ESCRIBIR_PALABRA
;  Escribe una palabra (caracter/atributo) en la memoria de
;  video. Si existe un adaptador de color, debe esperar por
;  un retrazo horizontal.
;

ESCRIBIR_PALABRA:

    cmp     adaptador,1             ;¿ Adaptador de Color ?
    jne     w_w                     ;No, escriba el caracter
    push    dx                      ;Si,entonces debe esperar
    mov     dx,3DAh                 ;por el retrazo horiz.

horizontal_test:

    in      al,dx
    test    al,1                    ;¿ Fin de un retrazo
                                    ;horizontal ?
    jnz     horizontal_test         ;No, espere
    cli                             ;Si, detectar el
                                    ;siguiente retrazo.
esperar:

    in      al,dx
    test    al,1                    ;¿ Inicio de un retrazo
                                    ;horizontal ?
    jz      esperar                 ;No, espere
    pop     dx                      ;Si, entonces...

w_w:

    mov     ax,bx                   ;Toma un caracter y lo
    stosw                           ;escribe diréctamente a
    sti                             ;la memoria de video.
ret
```

```
;
;                          ESCRIBIR_BYTE
;              Escribe una caracter en la memoria de video.
;

ESCRIBIR_BYTE:

    cmp     adaptador,1             ;¿ Adaptador de Color ?
    jne     w_b                     ;No, escriba caracter
    push    dx                      ;Si, entonces esperar por
    mov     dx,3DAh                 ;el retrazo horizontal.
hz_tst:
    in      al,dx
    test    al,1                    ;¿ Fin de un retrazo
                                    ;horizontal ?
    jnz     hz_tst                  ;No, espere
    cli                             ;Si, detectar siguiente
esp:
    in      al,dx                   ;retrazo.
    test    al,1                    ;¿ Inicio de un retrazo
                                    ;horizontal ?
    jz      esp                     ;No, espere
    pop     dx                      ;Si, entonces...
w_b:
    mov     al,bl                   ;Toma caracter y lo
    stosb                           ;escribe diréctamente a
    sti                             ;la memoria de video
    inc     di
    ret

;
;                MOSTRAR_TEXTO: Muestra un texto
;
;   Entrada: DX: Dirección de video para el texto
;            SI: Dirección del texto
;

MOSTRAR_TEXTO:

    call    fijar_cursor            ;Fija el cursor según DX
    cld
obtener_texto:
    lodsb                           ;¿ Toma caracter del Texto
    cmp     al,0                    ;Fin del texto ?
    jz      fin_text                ;Si, salir
    call    escribir_texto          ;No, mostrar el caracter
    jmp     short obtener_texto     ;Continuar
fin_text:                           ;retorno
    ret

ESCRIBIR_TEXTO:
    push    si                      ;Guarda SI
```

```
    mov     ah,14                   ;Subroutina para mostrar
    int     10h                     ;caracter en la memoria
                                    ;de video.
    pop     si                      ;Recupera SI
    ret

;
;   DESACTIVAR_CGA y ACTIVAR_CGA
;   Estas subroutinas desactivan o activan la salida de video
;   en una CGA. (Color Graphics Adapter).
;

DESACTIVAR_CGA:

    mov     dx,3DAh                 ;Dirección del Registro
esperar_retrazo:                    ;de Estado.
    in      al,dx                   ;Lee el estado.
    test    al,8                    ;¿Retrazo vertical activo?
    je      esperar_retrazo         ;No, entonces esperar
    sub     dx,2                    ;Dirección de MSR en DX
    mov     al,25h                  ;Valor para desactivar el
                                    ;video.
    out     dx,al                   ;Desactiva la salida de
    ret                             ;video.

ACTIVAR_CGA:

    mov     ah,15                   ;Toma el modo de video
    int     10h
    lea     bx,valores_CGA          ;Valores para habilitar
                                    ;el video.
    xlat                            ;Valor en AL
    mov     dx,3D8h                 ;Dirección del MSR.
    out     dx,al                   ;Activa salida de video.
    ret

;
;   ALERTA
;   Muestra algún mensaje de error, suena el parlante y
;   por el ingreso desde el teclado.
;

ALERTA:

    call    mostrar_texto           ;Muestra mensaje de error
    call    parlante                ;Suena la alarma.
    call    tomar_tecla             ;Toma un caracter desde
                                    ;el teclado.
    call    borrar_msg              ;Borra mensaje mostrado.
    call    ocultar_cursor          ;y oculta el cursor
    ret
```

```
;
;                          PREPARAR_NOMBRE
;  Esta subroutina almacena el nombre del archivo en el
;  formato DOS. Es decir, nombre_de_archivo, número_de_bytes
;  y fecha_hora de creación del archivo.
;

PREPARAR_NOMBRE:

     lea      si, dta+30        ;Posición del nombre del archivo
                                ;tomado desde el disco.
     mov      cx,12             ;12 bytes por cada nombre de
                                ;archivo, incluída su extensión.
sgte_almacenar:
     lodsb                      ;Toma un byte.
     cmp      al,0              ;¿ Fin del nombre.extensión del
                                ;archivo ?
     jz       fin_almacenar     ;Si, completar con espacios en
                                ;blanco.
     cmp      al,'.'            ;¿ Inicio de la extensión del
                                ;nombre ?
     jnz      almacenar_byte    ;No, almacene byte.
     sub      cx,3              ;Resta los tres bytes de la
                                ;extensión,
     mov      al,32             ;y completa el nombre del archivo
     rep      stosb             ;con espacios en blanco.
     add      cx,3              ;3 bytes para la extensión.
     jmp      short sgte_almacenar  ;Continúa para chequear la
almacenar_byte:                 ;extensión.
     stosb                      ;Almacena caracter del nombre.ext
                                ;del archivo.
     loop     sgte_almacenar    ;Y continúa con el siguiente
                                ;caracter.

;  Se alcanzó el final del nombre.extensión del nombre del
;  archivo por lo tanto si éste contiene menos de 12
;  caracteres, se completan los bytes restantes con espacios
;  en blanco.

fin_almacenar:
     mov      al,32             ;Caracter de espacio en blanco.
     rep      stosb             ;Completa bytes restantes.

; Ahora calcula la longitud del archivo.

longitud_archivo:
     push     di                        ;Guarda puntero
     add      di,8
     mov      dx,ds:[dta+26]            ;Toma valor MSB de la
                                        ;longitud del archivo.
     mov      ax,ds:[dta+28]            ;Valor LSB
     mov      bx,10                     ;Convierte a valor dec.
                                        ;divide para 10
     std                                ;Dirección inversa
```

```
sgte_longitud:
    mov     cx,dx                   ;Valor LSB en CX
    xor     dx,dx                   ;Borra MSB
    div     bx                      ;Convierte a decimal
    xchg    ax,cx                   ;Recupera valor LSB
    div     bx                      ;Convierte a decimal
    xchg    ax,dx                   ;Recupera el resto
    add     al,'0                   ;Convierte a valor ASCII
    stosb                           ;y lo almacena
    mov     ax,cx
    or      cx,dx                   ;¿Algo más por convertir?
    jnz     sgte_longitud           ;Si, dividir otra vez
    cld
    pop     di                      ;Recupera puntero
    add     di,11                   ;y lo fija al campo de
                                    ;fecha.

; Luego calcula la fecha de creación del archivo.

fecha:
    mov     dx,ds:[dta+24]          ;Toma valor de fecha
    mov     ax,dx
    mov     cl,5                    ;Desplaza los bits LSB
    ror     ax,cl
    and     ax,0fh                  ;Deja el mes
    mov     cl,0ffh
    mov     ch,'-'                  ;Caracter delimitante
    call  almacenar_palabra         ;Almacena

    mov     ax,dx                   ;Toma la fecha
    and     ax,1fh                  ;deja el dia
    mov     cl,0
    mov     ch,'-'
    call  almacenar_palabra         ;Almacena

    mov     ax,dx                   ;Toma la fecha
    mov     cl,9
    ror     ax,cl
    and     ax,7Fh                  ;Deja el año
    add     ax,80                   ;Ajusta a valor ASCII
    cmp     ax,100                  ;¿ Pasado del año 2000 ?
    jb      mostrar_fecha           ;No, mostrar
    sub     ax,100                  ;Si, entonces ajusta para
                                    ;el siguiente siglo.
mostrar_fecha:

    mov     cl,0
    mov     ch,32
    call  almacenar_palabra         ;Almacena
tiempx:
    inc     di                      ;Hora
    mov     dx,ds:[dta+22]          ;Toma la hora
    mov     ax,dx
    mov     cl,11
    ror     ax,cl
```

```
    and     ax,1Fh                  ;Deja las horas
    push    ax
    cmp     ax,12                   ;¿ Pasado del medio dia ?
    jbe     meridiano
    sub     ax,12                   ;Si, ajuste
meridiano:

    cmp     ax,0                    ;¿ Media noche ?
    jnz     no_media_noche
    mov     ax,12                   ;Si, ajuste
no_media_noche:

    mov     cl,0ffh
    mov     ch,':'
    call  almacenar_palabra         ;Almacena

    mov     ax,dx
    mov     cl,5
    ror     ax,cl
    and     ax,3Fh                  ;Deja los minutos
    mov     cl,0
    pop     dx                      ;Recupera las horas
    mov     ch,'p '                 ;Asume 'pm'
    cmp     dx,12                   ;¿ Estamos en 'pm' ?
    jae     pm
    mov     ch,'a '                 ;No, entonces cambia a
                                    ;'am'
pm: call  almacenar_palabra         ;Almacena
    mov     al,'H'
    test    byte ptr ds:[dta+21],6  ;¿ Es un archivo oculto
                                    ;o archivo del sistema ?
    jz      no_oculto
    stosb                           ;Si, fije una 'H' al
    jmp     fin_pn                  ;final del registro.
no_oculto:
    inc     di
fin_pn:
    ret

ALMACENAR_PALABRA:

    div     bl                      ;Divide para 10
    add     ax,'00'                 ;Convierte a ASCII
    cmp     cl,0
    jz      almacenar_caracter
    cmp     al,'0'
    jnz     almacenar_caracter
    mov     al,32

almacenar_caracter:

    stosw                           ;Almacena caracter
    mov     al,ch                   ;y caracter delimitador
    stosb
    ret
```

```
;
;  Las siguientes subroutinas borran las ventanas y áreas de
;  textos.
;

;  Borra la ventana de video para mostrar archivos de
;  directorio.

BORRAR_VENTANA_DIR:

    mov     bh,attr_de_dir          ;Monoc-07, Color-70h
    mov     cx,0101h                ;Desde línea 1, columna 1
    mov     dx,1729h                ;Hasta línea 23, colum 41
    jmp     short borrar_area       ;Borrar la ventana.

; Borra toda el área de video.

BORRAR_AREA_DE_VIDEO:

    mov     bh,attr_de_ver          ;Monoc:07 Color: 0Fh
    xor     cx,cx                   ;Desde línea 0, columna 0
    mov     dx,174fh                ;Hasta línea 23,colum. 79
    jmp     short borrar_area       ;Borrar.

; Borra la ventana de configuración.

BORRAR_VENTANA_CNFG:

    mov     bh,40h
    mov     cx,0718h                ;Desde línea 7, colum. 24
    mov     dx,1038h                ;Hasta línea 16,colum. 56

;  Borra el área de video, especificada en los registros  BH,
;  CX y DX.

BORRAR_AREA:

    mov     ax,600h                 ;Función del BIOS para
    int     10h                     ;borrar un sector de la
    ret                             ;memoria de video.

;  Borra el área de memoria reservada para almacenamiento  de
;  textos

BORRAR_AREA_TXT:

    lea   di,inicio_area_de_textos  ;Fija el cursor en la
    mov     cursor_texto,di         ;posición inicial del
                                    ;área de textos.
    mov     inicio_pagina,di        ;Inicio de ventana.
    mov     cx,3900+39              ;101 líneas, 78 columnas
                                    ;a borrar.
    mov     ax,' '                  ;Espacios en blancos.
    rep     stosw                   ;Borra área de memoria
```

```
    ret                                 ;direccionada por ES:DI

;
;                          BORRAR
;   Subroutina invocada cuando se desea borrar el contenido
;   de un texto en el EDITOR.
;

BORRAR:

    call    final_de_texto          ;¿Existe algo que borrar?
    jc      fin_borrar              ;No, entonces ignore
                                    ;pedido.
    lea     si, menu13              ;Si, enviar mensaje:

    call    mostrar_menu            ;'¿Borrar Texto? (S/N) _'
    mov     dx,1314h                ;Fija y activa el cursor
    call    fijar_cursor            ;en la línea 19, colum.20
    call    siono                   ;¿ SI o NO ?
    pushf
    call    ocultar_cursor          ;Oculta el cursor
    mov     si,menu
    call    mostrar_menu
    popf
    jc      fin_borrar              ;No, ignore pedido
    push    cs                      ;Si, ES=CS
    pop     es
    cld                             ;DI y SI se incrementarán
    call    borrar_area_txt         ;Borra el área de textos
    lea     di,menu2+9              ;Luego borra el nombre
    mov     cx,18                   ;del texto editado y
    mov     ax,' '                  ;colocado en MENU2+9.
    rep     stosw
    call    info                    ;Escribe MENU2 en la
fin_borrar:                         ;primera línea.
    ret

;
;          BORRAR_MSG: Borra un mensaje.
;

BORRAR_MSG:
    mov     di,direc_video          ;ES:DI a la posición
    add     di,17*160+43*2          ;donde existe el mensaje
    call    borrar_linea_msg        ;Borra una línea de
                                    ;mensajes.
    add     di,43*2                 ;Línea siguiente.
    call    borrar_linea_msg        ;Borra línea
    add     di,43*2                 ;Ultima línea
borrar_linea_msg:
    mov     es,sgmnto_de_video
```

```
    mov     cx,37                   ;74 caracteres a borrar
    mov     bl,' '                  ;Espacios en blanco
cm1:
    call    escribir_byte           ;Escribe caracter
    loop    cm1                     ;Lazo que se repite 74
                                    ;veces
    push    cs                      ;Recupera ES
    pop     es
    ret

;
;  MAS: Muestra el resto de opciones proporcionadas  por el
;  Editor.
;

MAS:
    lea     si,menu0
    cmp     si,menu                 ;¿Está mostrado el primer
                                    ;menú ?
    jne     nuevo_menu              ;No, muéstrelo
    lea     si,menu1                ;Si, entonces mostrar el
nuevo_menu:                         ;segundo menú.
    mov     menu,si                 ;Menú actual
    call    mostrar_menu            ;Muestra nuevo menú de
    ret                             ;opciones.

;
;                    ACTUALIZAR_RELOJ
;  Muestra en la ventana del editor la hora actual  en modo
;  contínuo. Este  reloj  solo  corre  cuando  el  programa
;  residente está activado.
;

ACTUALIZAR_RELOJ:

    push    cs                      ;ES al segmento del
    pop     es                      ;programa.
    sti
    dec     contador                ;¿ Pasó un segundo ?
    jz      reloj1                  ;Si, actualize el reloj
    ret                             ;No, entonces salir
reloj1:
    mov     contador,18             ;1 segundo
    mov     cl,60                   ;Divisor segundos/minutos
    mov     dx,0C0Ah
    lea     di,reloj[6]             ;Hora actual en valor
                                    ;ASCII
    mov     al,tiempo_actual.sec    ;Toma valor del segundo,
    inc     al                      ;lo incrementa
    cbw                             ;AH=0
    div     cl                      ;Divide segundos para 60
```

```
    mov     tiempo_actual.sec,ah    ;Residuo a segundo
    call    binario_a_ascii         ;Convierte seg. a valor
                                    ;ASCII y lo almacena en
                                    ;RELOJ_TEXT
    mov     al,tiempo_actual.min    ;Ahora toma el valor del
    add     al,ch                   ;minuto.
    cbw
    div     cl
    mov     tiempo_actual.min,ah
    call    binario_a_ascii
    mov     al,tiempo_actual.hrs    ;Finalmente toma el valor
                                    ;de la hora.
    add     al,ch
    mov     tiempo_actual.hrs,al
    cbw
    div     dh
    cmp     ah,0
    jne     cont
    mov     ah,12
cont:
    call    binario_a_ascii
    mov     reloj[9],'p'            ;Asume 'p'
    cmp     ch,1                    ;Estamos en 'p'
    je      mostrar_hora
    mov     reloj[9],'a'            ;No, cambie a 'a'
mostrar_hora:
    mov     es,sgmnto_de_video      ;ES al segmento de video
    mov     di,direc_video          ;Posición donde mostrará
    add     di,35*2                 ;el reloj.
    lea     si,reloj                ;Posición interna del
                                    ;reloj
    mov     bh,attr_del_reloj       ;Monoc-70h, Color-30h
    mov     cx,11                   ;11 caracteres
    cld
    call    mover                   ;Muestra los caracteres
    ret

;
;   FECHA_Y_HORA
;   Calcula y muestra la fecha y hora actual
;

FECHA_Y_HORA:

    mov     ah,2ch                  ;Obtiene la hora del dia
    int     21h
    mov     tiempo_actual.hrs,ch    ;Horas
    mov     tiempo_actual.min,cl    ;Minutos
    mov     tiempo_actual.sec,dh    ;Segundos
    mov     contador,1              ;Contador de los impulsos
                                    ;del reloj

    mov     es,sgmnto_de_video
```

```
        mov     di,direc_video
        add     di,160*13+32*2          ;Posición donde muestra
        mov     bl,'┤'                  ;la fecha.
        call    escribir_byte
        mov     ah,2ah                  ;Toma en CX:DX la fecha
        int     21h                     ;actual.
        push    cx                      ;Guarda fecha actual
        push    dx
        lea     si,dias                 ;Días de la semana
        call    buscar                  ;Calcula y muestra el día
                                        ;de la semana.
        pop     ax                      ;Recupera la fecha
        push    ax
        call    mostrar_lc              ;Día del mes
        mov     bx,7020h                ;Espacio
        call    escribir_palabra
        lea     si,meses                ;Meses del año
        pop     ax
        mov     al,ah
        dec     al
        call    buscar                  ;Calcula y muestra el mes
        pop     ax                      ;Finalmente toma el valor
        call    bascii4                 ;del año y lo muestra
        mov     bl,'├'                  ;Delimitador
        call    escribir_byte
        ret

BUSCAR:

        mov     bl,3                    ;3 caracteres por día o
                                        ;mes.
        xor     ah,ah                   ;Calcula la posición del
                                        ;día o mes actual.
        mul     bl
        add     si,ax
        mov     cx,3
        mov     bh,lincol_attr
        call    mover                   ;Muestra el día o el mes
        mov     bx,7020h                ;y deja un espacio en
        call    escribir_palabra        ;blanco.
        ret


;
;   Subroutina para convertir un byte a valor ASCII.
;

BINARIO_A_ASCII:

        mov     ch,al
        mov     al,ah
        cbw
        div     dl
        add     al,48
```

```
    mov     [di],al
    inc     di
    add     ah,48
    mov     [di],ah
    sub     di,4
    ret


;
;   Subroutina para convertir 4 bytes a valor ASCII.
;

BASCII4:

    mov     cx,0010
    lea     si,valor_ascii+3
c20:
    cmp     ax,0010
    jb      c30
    xor     dx,dx
    div     cx
    or      dl,30h
    mov     [si],dl
    dec     si
    jmp     c20
c30:
    or      al,30h
    mov     [si],al
    mov     cx,4
    cld

MOVER:

    lodsb
    mov     bl,al
    call    escribir_palabra
    loop    mover
    ret


;
;                          CONFIGURAR
;   Permite configurar el Puerto de comunicación  usado  por
;   la línea de TELEX, el Directorio de  mensajes enviados o
;   recibidos y el INDICATIVO del abonado.
;

CONFIGURAR:
    call    ventana_cnfg            ;Forma la ventana de
                                    ;Configuración.
    call    actualizar_cnfg         ;Muestra Configuración
                                    ;actual
paso1:                              ;Toma un caracter desde
```

```
    call    tomar_tecla                 ;el teclado
    cmp     al,ESC                      ;¿ ESC ?
    jne     paso10
    call    ocultar_cursor              ;Si, ocultar el cursor y
    call    actualizar_directorio_tlx ;salir.
    ret
paso10:
    cmp     ah,50h                      ;¿ Bajar de campo ?
    je      paso2                       ;Si,
    cmp     al,ENTER                    ;¿ INTRO ?
    jne     paso3                       ;No, chequear subida
paso2:
    cmp     campo,1                     ;¿ Ultimo campo ?
    jne     bajar_de_campo              ;No, bajar
    mov     campo,-1                    ;Si, colocar al primer
bajar_de_campo:                         ;campo.
    inc     campo                       ;Campo siguiente
    call    actualizar_cnfg             ;Muestra Campo siguiente
    jmp     paso1                       ;Tomar nueva tecla
paso3:
    cmp     ah,48h                      ;¿ Subir de campo ?
    jne     paso4                       ;No, chequear caracteres
    cmp     campo,0                     ;Si, ¿ Primer campo ?
    jne     subir_de_campo              ;No, subir de campo
    mov     campo,2                     ;Si, colocar en el último
subir_de_campo:                         ;campo.
    dec     campo                       ;Campo siguiente
    call    actualizar_cnfg             ;Muestra Campo
    jmp     paso1                       ;Tomar nueva tecla
paso4:
    cmp     campo,0                     ;¿ Campo de 'DIR TELEX' ?
    jne     paso6                       ;Si es igual, se evita el
    cmp     al,' '                      ;espacio en blanco
    je      paso1
paso6:
    cmp     al,8                        ;¿ Retroceder ?
    jne     sgte_cnfg                   ;No, chequear caracteres
    xor     al,al                       ;Si,
    dec     puntero_cnfg                ;Disminuye puntero de
    jmp     paso7                       ;caracteres.
sgte_cnfg:
    cmp     al,' '                      ;¿ Caracter menor a 20h ?
    jb      paso1                       ;Si, ignórelo.
    cmp     al,'~'                      ;Superior a '~'
    ja      paso1                       ;Si, ignórelo
paso7:
    mov     si,puntero_cnfg             ;Posición para caracter
                                        ;ingresado
    mov     byte ptr [si],al            ;Guarda caracter.
    call    actualizar_cnfg             ;Muestra cambio de
                                        ;configuración
    jmp     paso1                       ;Toma nueva tecla

campo_actual:
    mov     di,160*13+26*2-1            ;Posición de video del
```

```
                                        ;primer campo
    mov     bx,160*2                    ;2 líneas
    mov     al,campo                    ;Campo actual
    xor     ah,ah                       ;Calcula cuántas líneas
    xor     dx,dx                       ;se deben agregar a DI
    mul     bx                          ;para fijarse en la
                                        ;memoria de video del
    add     di,ax                       ;campo activo
    lea     si,campo                    ;Ahora calcula el número
    lodsb                               ;de caracteres que serán
    xor     ah,ah                       ;invertidos en el campo.
    add     si,ax
    lodsb
    mov     cl,al                       ;Número de caracteres por
    xor     ch,ch                       ;invertir.
attr_campo:
    mov     al,70h                      ;Atributo para invertir
    cld                                 ;caracteres.
sgte_attr:
    stosb                               ;Cambia el atributo.
    inc     di                          ;Siguiente caracter.
    loop    sgte_attr                   ;Repite CX veces.
    ret

ACTUALIZAR_CNFG:

    lea     si,port                     ;Campo 'Puerto Serial'
    mov     di,160*9+24*2
    mov     bh,4Fh                      ;Atributo de líneas
    mov     atributo,1
    mov     puntero_de_linea,4          ;4 líneas
linea_sgte:
    mov     cx,33                       ;33 caracteres por línea
    call    mostrar_linea               ;Muestra las líneas
    add     di,254                      ;47*2+160
    dec     puntero_de_linea            ;¿ Alguna línea más por
                                        ;mostrar ?
    jnz     linea_sgte                  ;Si, continúe
    call    campo_actual                ;Muestra campo actual
    mov     di,160*9+53*2-1             ;Campo 'Ningún puerto
                                        ;serial'
    cmp     puerto_serial,0             ;¿ Existe algún puerto
                                        ;serial instalado ?
    jz      puerto_actual               ;No, invierta este campo
    sub     di,12                       ;Campo 'COM2'
    cmp     puerto_serial,2F8h          ;¿ Telex en COM2 ?
    jz      puerto_actual               ;Si, invierta este campo
    sub     di,12                       ;No, entonces telex usa
puerto_actual:                          ;COM1.
    mov     cx,4
    call    attr_campo                  ;Muestra puerto serial
                                        ;usado por TELEX
    mov     di,160*11+35*2-1            ;Campo 50 bauds
    cmp     bauds,2304                  ;¿ Telex funcionando a 50
                                        ;baudios ?
```

```
    je      bauds_actuales          ;Si, invierta este campo
    add     di,10                   ;Campo 70 bauds
    cmp     bauds,1646              ;¿ Telex funcionando a 70
                                    ;baudios ?
    je      bauds_actuales          ;Si, invierta este campo
    add     di,10                   ;No, entonces TELEX está
bauds_actuales:                     ;operando a 100 baudios.
    mov     cx,3
    call    attr_campo              ;Muestra la velocidad de
                                    ;transmisión a la que el
                                    ;TELEX está funcionando
    mov     dx,0D25h                ;Posible posición del
                                    ;cursor
    lea     si,dir_telex            ;SI en el campo de
                                    ;'DIR TELEX'
    cmp     campo,0                 ;¿ Campo de 'DIR TELEX' ?
    je      columna_del_cursor      ;No, entonces modifica
    add     dh,2                    ;DH y SI si está activo
    lea     si,indicativo           ;el campo del indicativo
columna_del_cursor:
    cmp     byte ptr [si],0         ;¿ Fin del campo ?
    je      cursor_cfg              ;Si, chequear espacio
    inc     dl                      ;disponible.
    inc     si
    jmp     columna_del_cursor
cursor_cfg:
    cmp     dl,38h                  ;¿ Hay espacio disponible
                                    ;en el campo ?
    jb      actlz_cursor            ;Si, actualiza el cursor
    dec     dl                      ;No, entonces...
    dec     si                      ;Borra caracter ingresado
    mov     byte ptr [si],0         ;anteriormente
actlz_cursor:
    mov     puntero_cnfg,si         ;Posición para siguiente
                                    ;caracter
    call    fijar_cursor            ;Actualiza el cursor
    ret                             ;y retorna
fin_actualizar:
    call    ocultar_cursor          ;Oculta el cursor
ret
actualizar_directorio_tlx:
    push    cs                      ;ES al segmento del
    pop     es                      ;programa
    lea     si,dir_telex            ;Directorio del telex
    lea     di,refile               ;Cadena:
    cld                             ;[dir_tlx][archivo_tlx]
    call    sgte_sendero            ;Transfiere directorio
                                    ;del telex
    cmp     byte ptr es:[di-1],'\'  ;Si es necesario, agrega
    je      actualizar_archivo      ;un '\'
    mov     al,'\'
    stosb
actualizar_archivo:
    lea     si,archivo_de_telex     ;Nombre de archivo para
                                    ;próximo mensaje recibido
```

```
    call    sgte_sendero            ;Transfiere el nombre
    stosb                           ;Finaliza con un 0
    ret

sgte_sendero:
    lodsb                           ;Toma caracter
    cmp     al,0                    ;¿ Fin de la cadena ?
    je      fin_sendero             ;Si, salir
    stosb                           ;No, entonces transferir
    jmp     sgte_sendero            ;caracter y continuar.
fin_sendero:
    ret

;
;       ENVIAR_CARACTER: Envía el caracter dado en AL
;

ENVIAR_CARACTER:
    mov     dx,puerto_serial        ;Dirección del puerto de
                                    ;salida (COM1/2)
    push    ax                      ;Guarda valor a enviar
esperar_envio:
    call    ck_estado_linea         ;Toma estado de la línea
    test    al,20h                  ;¿ Transmisor listo para
                                    ;enviar otro caracter ?
    jz      esperar_envio           ;No, espere
    pop     ax                      ;Si, recupera valor,
    out     dx,al                   ;y lo envía por el puerto
    ret                             ;serial.

;
;                   CK_ESTADO_LINEA
;       Chequea el estado de la linea de comunicación
;

CK_ESTADO_LINEA:

    add     dx,6                    ;Registro de Control del
                                    ;Modem
    in      al,dx                   ;Toma en AH el estado del
    mov     CTS_DSR,al              ;Modem.
    nop                             ;El acceso al 8250 debe
    dec     dx                      ;ser por lo menos después
                                    ;de 2 microsegundos.
    in      al,dx                   ;Toma en AL el estado de
                                    ;la línea, y deja DX
    sub     dx,5                    ;apuntando al puerto de
                                    ;datos.
    test    al,1                    ;Retorna con ZF=0 si un
    ret                             ;caracter está listo para
                                    ;ser tomado desde el UART
```

```
;
;                              MARCAR
;  Fija DTR y RST a nivel alto. Con esto, IETEL debe
;  la linea de telex.
;

MARCAR:

     test     en_linea,1              ;¿ Linea invertida ?
     jne      marcar_numero           ;Si, entonces marque el
                                      ;número del abonado.
     mov      dx,puerto_serial        ;DX en el registro de
                                      ;control de modem
     add      dl,4                    ;COM1:3FCh  COM2:2FCh
     mov      al,3                    ;DTR y RTS forzadas a
                                      ;nivel alto.
     out      dx,al                   ;Con esto, la linea de
     lea      dx,area_de_telex        ;TELEX debe invertirse
     mov      puntero_de_telex,dx
     mov      puntero_anterior,dx
     ret

;
;     MARCAR_NUMERO: Envia el número del abonado a llamar
;

MARCAR_NUMERO:

     call     ck_numero               ;¿Es correcto el número a
                                      ;llamar ?
     jc       fin_del_numero          ;No, salir
     lea    si,inicio_area_de_textos  ;Posición del número del
                                      ;abonado.
     cld                              ;Autoincremento
sgte_marcar:
     call     tecla
     cmp      ah,43h                  ;¿ F9 ?
     jne      ck_parar_envio
     jmp      colgar                  ;Si, cortar la llamada
ck_parar_envio:
     cmp      ah,41h                  ;¿ F7 ?
     jne      numero_sgte
     ret                              ;Si, salir
numero_sgte:                          ;No, continuar marcado
                                      ;del número.
     cmp      byte ptr [si-1],'+'     ;¿Fin número del abonado?
     je       fin_del_numero          ;Si, salir
     lodsb                            ;No, entonces toma el
     push     si                      ;siguiente digito,
     call     sacar                   ;lo envia,
     call     mostrar_caracter        ;y lo muestra en la
     pop      si                      ;pantalla.
```

```
    jmp     sgte_marcar             ;Chequea el teclado
fin_del_numero:
    ret


;
;  ┌──────────────────────────────────────────────────┐
;  │                    CK_NUMERO                     │
;  │          Interpreta el número del abonado        │
;  │                                                  │
;  │      Salida:   CY=0  Número correcto             │
;  │                CY=1  Error en el número          │
;  └──────────────────────────────────────────────────┘
;

CK_NUMERO:
    lea     si,inicio_area_de_textos ;El número del abonado
    mov     cx,78                   ;debe estar solo en la
    cld                             ;primera línea del texto
                                    ;y finalizado con el
sgte_byte:                          ;signo '+'.
    lodsb
    cmp     al,'+'                  ;¿ Fin del número ?
    jz      numero_OK               ;Si, número correcto
    loop    sgte_byte               ;No, continuar chequeando

numero_erroneo:
    call    parlante                ;El número es incorrecto,
    lea     si,menu16               ;por lo tanto suena el
    call    mostrar_menu            ;parlante y se envía un
    call    tomar_tecla             ;mensaje, luego de lo
    mov     si,menu                 ;cual, se actualiza la
    call    mostrar_menu            ;línea de menú.
    stc                             ;CY=1 indica el error
    ret
numero_OK:
    clc                             ;CY=0 indica que el
    ret                             ;número es correcto.


;
;  ┌──────────────────────────────────────────────────┐
;  │ ENVIAR: Envía un texto por la línea de comunicación │
;  └──────────────────────────────────────────────────┘
;

ENVIAR:
    call    final_de_texto          ;¿Existe algo por enviar?
    jc      salir_del_envio         ;No, salir
    cld
    call    context                 ;Si, comprimir el texto
    mov     si,dir_Grb_Imp_Env      ;Posición inicial texto
    cld                             ;Ahora busca el inicio de
                                    ;la segunda línea.
    mov     ax,0A0Dh                ;Códigos CR-LF
ck_inicio:
    cmp     ax, word ptr [si]       ;¿ Fin de línea ?
```

```
    je      bytes_a_enviar          ;Si, calcular los bytes a
                                    ;enviar
    inc     si                      ;No, compare posiciones
    jmp     ck_inicio               ;siguientes.
bytes_a_enviar:
    add     si,2
    lea     ax,nombres_de_archivos+8000
    cmp     ax,si                   ;¿Existe algo por enviar?
    jbe     salir_del_envio         ;No, salir
    sub     ax,si                   ;Si, entonces calcula el
    mov     num_bytes,ax            ;número de bytes que se
enviar_texto:                       ;enviarán.
    call    tecla                   ;Chequea el teclado
    cmp     ah,42h                  ;¿Parar el envio ? [F8]
    je      salir_del_envio         ;Si, salir
    cmp     ah,43h                  ;¿Colgar la llamada? [F9]
    jne     byte_a_enviar           ;No, continúe enviando
    jmp     colgar                  ;Si, colgar comunicación.
byte_a_enviar:
    lodsb                           ;Toma siguiente caracter,
    push    si
    call    sacar                   ;lo envía por la línea
    call    mostrar_caracter        ;y lo muestra en la
    pop     si                      ;ventana del telex.
    dec     num_bytes               ;¿ Algún caracter más por
                                    ;enviar ?
    jne     enviar_texto            ;Si, continúe.
salir_del_envio:                    ;No, salir
    ret

;
;   SACAR: Envía el caracter contenido en AL
;

SACAR:
    push    cx                      ;CX no debe modificarse
    push    ax
    call    ascii_baudot            ;Código en AL a código
    pop     bx                      ;BAUDOT.
    push    bx
    cmp     ax,bx                   ;¿ AL fue convertido ?
    je      no_se_envia             ;No, salir
    cmp     ah,0                    ;Si,
                                    ;¿Cambio Letras/números ?
    je      enviar_AL               ;No, enviar caracter de
                                    ;AL (Figures/letters)
    push    ax                      ;Si, entonces enviar AH
    mov     al,ah                   ;y luego el caracter de
    call    enviar_caracter         ;AL.
    pop     ax
enviar_AL:
    call    enviar_caracter         ;Envía el caracter de AL
no_se_envia:
```

```
    pop     bx
    mov     ax,bx                   ;Retorna sin modificar AX
    pop     cx                      ;Recupera CX.
    ret

;
;                          CK_ENTRADA_SERIAL
;  Chequea el estado del puerto de comunicación y  verifica
;  si existe un caracter disponible en la línea.
;

CK_ENTRADA_SERIAL:

    mov     dx,puerto_serial        ;Apunta DX al puerto de
                                    ;salida serial
    call    ck_estado_linea         ;Chequea la línea.
    jne     tomar_codigo            ;Si existe un caracter
ck_linea:                           ;disponible lo toma.
    add     dl,4                    ;DX en el registro de
                                    ;control del modem.
    test    en_linea,1              ;¿La línea está invertida?
    je      ck_inversion_de_linea   ;No, chequear ingreso de
                                    ;la línea
    test    CTS_DSR,10h             ;Si, ¿ está CTS en nivel
                                    ;alto ?
    jne     anule_la_linea          ;Si, anule bit de línea
                                    ;invertida
    dec     word ptr time_tlx       ;¿ Ha estado la línea
                                    ;invertida durante 3 min.
                                    ;sin envío o recepción
                                    ;alguna ?
    je      cortar_comunicacion     ;Si, cortar comunicación.
    ret                             ;No, salir.

cortar_comunicacion:
    mov     segundo,18
anule_la_linea:
    xor     al,al

    out     dx,al                   ;RTS y DTR forzadas a
                                    ;nivel bajo.
    and     en_linea,0              ;Indicador de línea no
                                    ;invertida.
    or      cs:bits_tlx,100b        ;Fija bit de grabado de
    mov     indicador_tlx,' '       ;texto recibido.
    ret
ck_inversion_de_linea:
    test    CTS_DSR,10h             ;¿ CTS en nivel bajo ?
    je      poner_en_linea          ;Si, fije bit de línea
    ret                             ;invertida.
poner_en_linea:
    cmp     segundo,0               ;RTS y DTR se invierten
    je      comenzar_a_fijar        ;1 seg. después de haber
```

```
    dec     segundo                 ;detectado la inversión
    ret                             ;de CTS.
comenzar_a_fijar:
    mov     al,3
    out     dx,al                   ;RTS y DTR forzadas a
                                    ;nivel alto.
    or      en_linea,1              ;Indicador para linea
    mov     indicador_tlx,'C'       ;invertida.
    call    parlante
    ret
tomar_codigo:
    mov     time_tlx,18*60*3        ;3 minutos
    in      al,dx                   ;Recibe caracter en
                                    ;código BAUDOT.
    cmp     lfsh_in,26              ;¿ Se están recibiendo
                                    ;letras ?
    jne     ck_baudot               ;Si, chequee sus códigos
    cmp     al,9                    ;¿Código de WHO ARE YOU?
                                    ;(¿Quién es Usted ?)
    jne     ck_conv                 ;No, chequea BELL
    jmp     here_is                 ;Si, envía el indicativo
ck_conv:
    cmp     al,11                   ;¿ Código de BELL ?
                                    ;(Parlante)
    jne     ck_baudot               ;No, continúe
    jmp     parlante                ;Si, sonar el parlante
ck_baudot:
    call    baudot_ascii            ;Convierte el código
    jnc     mostrar_caracter        ;recibido al código ASCII
    ret

;
;   MOSTRAR_CARACTER
;   Muestra en la ventana de telex el caracter recibido
;

MOSTRAR_CARACTER:
    mov     si,puntero_de_telex     ;Almacena el caracter
                                    ;recibido en el área de
    mov     byte ptr ds:[si],al     ;textos recibidos.
    inc     puntero_de_telex        ;Incrementa puntero.
    sub     si,24
    call    ventana_telex           ;Ventana de telex
    mov     di,54*2+160             ;Posición de la ventana
                                    ;de video.
    mov     cx,25                   ;25 caracteres a mostrar
    cld
sh1:
    lodsb                           ;Toma un caracter
    mov     bh,70h                  ;Atributo del caracter.
    cmp     al,13                   ;¿ Código de CR ?
    jne     sh2                     ;No, chequear LF
    mov     al,'M'                  ;Si, muestre una 'M' con
```

```
    mov     bh,7                    ;fondo invertido.
    jmp     sh3
sh2:
    cmp     al,10                   ;Código de LF
    jne     sh3                     ;No, muestre caracter
    mov     al,'J'                  ;Si, muestre una 'J' con
    mov     bh,7                    ;fondo invertido
sh3:
    mov     bl,al                   ;Deposita en BL el
                                    ;caracter a mostrar
    call    escribir_palabra        ;Muestra el par caracter
                                    ;con su atributo.
    loop    sh1                     ;Repite la secuencia
    ret                             ;25 veces.
```

```
;
;                          BAUDOT_ASCII
;          Convierte del código BAUDOT al código ASCII
;
;   Entrada:                    Salida:
;   AL=Caracter BAUDOT      AL =  Caracter ASCII
;                           CY =1 Indica cambio de LSH-FSH
;
```

```
BAUDOT_ASCII        PROC
    xor     bx,bx
    cmp     al,lsh                  ;¿ Letters Shift ?
    je      numeros                 ;Si, fijar CY=1
    cmp     al,fsh                  ;¿ Figures Shift ?
    jne     ck_BDT                  ;No, chequear código
    mov     bx,26                   ;Baudot.
numeros:
    mov     lfsh_in,bx              ;Cambio de Letras a
    stc                             ;Figuras o viceversa
    ret
ck_BDT:
    lea     si,ascii                ;Fija SI en la posición
    add     si,lfsh_in              ;ASCII de Letras a
                                    ;Letras o Figuras según
                                    ;valor en LFSH
    lea     di,baudot               ;Cadena caracteres Baudot
    mov     cx,26                   ;26 comparaciones
    call    convertir               ;Convertir BAUDOT a ASCII
    jne     crlf_in                 ;¿ Tampoco ?.. Chequear
AL_ascii:                           ;CR LF.
    clc                             ;Indicador caracter
    ret                             ;recibido.

; Chequea envio de caracteres CR, LF, y espacios en blanco  e
; invertido
; DI = OFFSET BAUDOT+26
; SI = OFFSET ASCII+52
```

```
crlf_in:
    lea     si,ascii+52             ;SI en posición códigos
                                    ;ASCII de CR, LF, SB
    mov     cx,4                    ;4 comparaciones
    call    convertir               ;Convertir caracter
 jmp        short AL_ascii          ;retorno

BAUDOT_ASCII        ENDP


;
;  ASCII_BAUDOT
;       Convierte del código ASCII al código BAUDOT
;
;  Entrada:                 Salida:
;  AL = Caracter ASCII      AL = Caracter BAUDOT
;                           AH = Indica cambio de LSH-FSH
;
;  Si el caracter no es convertido AX retorna con  su valor
;  inicial.
;

ASCII_BAUDOT        PROC
    push    ax                      ;Guarda para después
    and     al,5fh                  ;Convierte a mayúscula
    lea     di,ascii                ;Cadena caracteres ASCII
                                    ;(Letters Shift)
    lea     si,baudot               ;Cadena caracteres BAUDOT
    mov     cx,26                   ;26 comparaciones.
    call    convertir               ;Convertir caracter ASCII
                                    ;a BAUDOT.
    jne     ck_fsh                  ;Si no fue convertido,
                                    ;chequea caracteres de
                                    ;Figures Shift.
    cmp     lfsh_out,0              ;¿ El envio del caracter
                                    ;anterior fue una letra ?
                                    ;A(a) - Z(z)
    je      AL_baudot               ;Si, entregar el caracter
                                    ;convertido.
    mov     lfsh_out,0              ;No, entonces se entregan
                                    ;dos caracteres:
    mov     ah,lsh                  ;El caracter de Letters
                                    ;Shift en AH y el
                                    ;caracter convertido en
    jmp     short AL_BAUDOT         ;AL.

; Verifica el envio de caracteres Figures Shift:
;      ;1234567890-?:(),=+ WHY BELL /.
; DI = OFFSET ASCII+26
; SI = OFFSET BAUDOT

ck_fsh:
    pop     ax                      ;Recupera el caracter a
    push    ax                      ;convertir y lo vuelve a
```

```
                                        ;guardar.
    mov     cx,26                       ;26 nuevas comparaciones
    call    convertir                   ;Convertir caracter
    jne     ck_crlf                     ;Si no fue convertido,
                                        ;chequea caracteres CR,
                                        ;LF, y espacios en blanco
                                        ;e invertido.
    cmp     lfsh_out,26                 ;¿ El envío del caracter
                                        ;anterior fue un valor
                                        ;numérico ?
    je      AL_baudot                   ;Si, entregar el caracter
                                        ;convertido y AH=0.
    mov     lfsh_out,26                 ;No, entonces se entregan
                                        ;dos caracteres:
    mov     ah,fsh                      ;El caracter de Figures
                                        ;Shift en AH y caracter
    jmp     short AL_baudot             ;convertido en AL

; Chequea el envio de caracteres CR, LF, y espacios en blanco
; e invertido.
; DI = OFFSET ASCII+52
; SI = BAUDOT+26

ck_crlf:
    pop     ax                          ;Recupera AX
    push    ax                          ;y lo guarda
    add     si,26                       ;SI posición de códigos
                                        ;BAUDOT de CR, LF, SB, SI
    mov     cx,4                        ;4 comparaciones
    call    convertir                   ;Convertir caracter
    je      AL_baudot                   ;Si no fue convertido, AX
    pop     ax                          ;retorna con su valor.
    ret
AL_baudot:
    pop     bx
    ret
ASCII_BAUDOT        ENDP

;
;   ┌──────────────────────────────────────────────────────────┐
;   |                        CONVERTIR                         |
;   | Convierte de código ASCII a código BAUDOT o viceversa    |
;   | según valor en SI y DI.                                  |
;   |                                                          |
;   | Entrada:                    Salida:                      |
;   | AL: Caracter a convertir    AL: Caracter convertido      |
;   | CX: Número de comparaciones ZF=0 Indica que el caracter  |
;   |                                  no pudo ser convertido  |
;   └──────────────────────────────────────────────────────────┘

CONVERTIR:
    push    cs
    pop     es                          ;ES al segmento del
    assume  es:code                     ;programa.
```

```
        jnz     fin_lineas
        inc     si
fin_lineas:
        ret


TABX:
        push    cx                      ;Guarda contador
        dec     cx
        and     cx,7
        inc     cx                      ;Ajusta
        push    cx
        call    llenar_espacios         ;Siguiente posición de
        pop     ax                      ;TAB.
        pop     cx
        sub     cx,ax                   ;Si no está en la columna
                                        ;80 toma el siguiente
        jnz     lineas_sgtes            ;caracter.
        ret

LLENAR_ESPACIOS:

        mov     al,32                   ;Caracter de espacio en
ck_display:                             ;blanco.
        cmp     indicador_video,1       ;¿ Estamos escribiendo en
                                        ;la pantalla ?
        jnz     ck_fin_disp             ;No, salir
        mov     bl,al
        push    bx
        mov     es,sgmnto_de_video
byte_a_video:
        call    escribir_byte           ;Si, escribe CX espacios
        loop    byte_a_video            ;en blanco.
        pop     bx
        push    cs                      ;Deja ES apuntando al
        pop     es                      ;segmento del programa
ck_fin_disp:
        ret                             ;y sale.


;
;   Las siguientes subroutinas controlan el paginado de la
;   visualización del contenido de un archivo.
;


HACIA_ADELANTE:

        sub ultima_pagina,cuatro_Kbytes ;Ajusta página actual
        xor     c: ,cx                  ;Mueve puntero de archivo
        mov     dx,cuatro_Kbytes        ;4 Kbytes hacia adelante
        call    mover_puntero

LEER_PRIMER_BLOQUE:
        lea     si, segundo_bloque
```

```
nombres_de_archivos = pc            ;Area para mostrar el
pc                  = pc + 8000     ;listado de los archivos
                                    ;de un directorio.
fin_area_dir        = pc            ;8000 bytes.
pc                  = pc + 127
area_ver_archivos   = pc            ;Area para mostrar el
pc                  = pc + 4096     ;contenido de un archivo.
segundo_bloque      = pc            ;8192 bytes.
pc                  = pc + 4096
dta                 = pc
pc                  = pc + 64 + 64  ;64 bytes para el DTA y
nueva_pila          = pc            ;64 bytes para la pila.
```

```
;
;   Inicialización
;   Prepara el programa para mantenerlo residente
;
```

```
initialize  proc    near
    assume  ds:code
errmsg2     db 13,10,'Bandera de Error Crítico no encontrada'
    db      13,10,'$'
errmsg3     db  13,10,
    db      'El programa ya está residente en memoria...'
    db      13,10,'$'
errmsg4     db      13,10,
    db      'Puertos Serie COM1 y COM2 no están presentes en'
    db      ' el sistema...',13,10,'$'
puerto_de_telex     db      13,10,'TELEX instalado en COM  '
    db      13,10,'$'
DOS_ERR     db 13,10,'Versión del DOS no soportada',13,10,'$'

; Chequea si el programa ya está residente en memoria.

INICIO:
    mov     word ptr [begin],0
    xor     bx,bx
    mov     ax,cs
init1:
    inc     bx
    cmp     ax,bx
    je      no_instalado
    mov     es,bx
    lea     si, begin
    mov     di,si
    mov     cx,16
    cld
    repe    cmpsb
    jne     init1

; Editor ya ha sido instalado.

    lea     dx,errmsg3
```

```
    mov     bx,cx
    cld                             ;Buscar hacia adelante.
    repne   scasb                   ;Busca caracter.
    jne     no_conv                 ;Sale con ZF=0 si no lo
                                    ;encuentra.
    sub     bx,cx                   ;Calcula la posición del
    add     si,bx                   ;caracter.
    dec     si
    lodsb                           ;Toma el caracter
                                    ;convertido,
    xor     ah,ah                   ;y retorna con ZF=1
no_conv:
    ret

;
;                          COLGAR
;  Fija las señales DTR y RTS a nivel bajo para anular la
;  comunicación.
;

COLGAR:
    mov     dx,puerto_serial        ;DX en el registro de
    add     dl,4                    ;control del modem.
    mov     al,0                    ;DTR y RTS forzadas a
                                    ;nivel bajo.
    out     dx,al                   ;Anula la comunicación
    ret

;
;          HERE_IS: El abonado contesta quién es
;

HERE_IS:
    call    enviar_crlf             ;Primero envía un CR y un
                                    ;LF.
    lea     si,indicativo           ;Indicativo
sgte_answ:
    lodsb                           ;Toma caracter del
    push    si                      ;indicativo,
    call    sacar                   ;lo envía,
    call    mostrar_caracter        ;y lo muestra
    pop     si
    cmp     byte ptr [si],0         ;¿ Fin del indicativo?
    jne     sgte_answ               ;No, continúe enviándolo
    ret

;
;          ENVIAR_CRLF: Envia un CR y un LF
;
```

```
ENVIAR_CRLF:
    mov     al,8                    ;CR
    call    enviar_caracter         ;Envia el caracter de
                                    ;Retorno del carro
    mov     al,13                   ;CR en ASCII
    call    mostrar_caracter
    mov     al,2                    ;LF
    call    enviar_caracter         ;Envia el caracter de
                                    ;Avance de Linea
    mov     al,10                   ;LF en ASCII
    call    mostrar_caracter
    ret

;
;                    MODO_CONVERSACIONAL
;  Permite visualizar la recepción de mensajes, así como
;  realizar diálogos entre dos abonados de la red TELEX .
;

MODO_CONVERSACIONAL PROC    NEAR
    push    cs
    pop     es                      ;ES al segmento del
    assume  es:code                 ;programa.

    push    cursor_texto
    push    inicio_pagina
    mov     attr_de_texto,0Fh       ;Atributo

; Guarda el texto que contiene el EDITOR.

    lea     si,inicio_area_de_textos
    lea     di,area_ver_archivos
    mov     cx,3900
    rep     movsw

; Borra  espacio  del EDITOR que será usado  por  el  mensaje
; recibido.

    call    borrar_area_txt
    lea     si,area_de_telex        ;Inicio del mensaje
    mov     cx,puntero_de_telex     ;recibido.
    mov     puntero_anterior,cx
    sub     cx,si                   ;¿ Algo recibido ?
    jcxz    inicio_dialogo          ;No, continúe.
                                    ;Si, entonces...

; Transfiere el mensaje recibido al área del EDITOR.

telex_al_editor:
    lodsb                           ;Toma caracter recibido
    call    cr_lf                   ;y lo pasa al EDITOR
    loop    telex_al_editor         ;CX caracteres
```

```
inicio_dialogo:
    lea     si,menu18               ;Muestra mensaje:
    call    info1                   ;'Mensaje Recibido...'
ver_mas:
    call    linea_columna           ;Línea y columna del
    mov     si,inicio_pagina        ;cursor.
    test    ds:bits_tlx,100b
    je      mostar_pagina
    call    grabar_mensaje
mostar_pagina:
    call    pagina                  ;Muestra 12*78 caracteres
                                    ;de texto.
    call    tecla                   ;Chequea el ingreso desde
                                    ;el teclado.
    cmp     al,ESC                  ;¿ ESC ?
    je      fin_modo_conversacional ;SI, salir
    cmp     al,1fh                  ;¿ Caracter >= a 20H ?
    jb      vm1
    call    pulsacion

    call    sacar                   ;Si, almacene y envíe el
    call    mostrar_caracter        ;caracter.
    jmp     short vm3
vm1:
    cmp     al,ENTER                ;¿ INTRO ?
    jne     vm2
    call    enviar_CRLF             ;Se envía un CR y un LF
    jmp     short vm3
vm2:
    call    opciones_del_telex      ;Chequea WHY, BELL , HRIS
    jc      vm3
    mov     cx,17                   ;17 opciones
    call    ck_menu                 ;Chequea teclas control
vm3:
    mov     si,puntero_anterior     ;del cursor.
    cmp     si,puntero_de_telex     ;¿ Algún otro caracter
                                    ;recibido ?
    je      ver_mas                 ;No, esperar
    lodsb                           ;Si, tomarlo desde área
                                    ;del telex y pasarlo al
    call    cr_lf                   ;EDITOR.
    inc     puntero_anterior        ;posición siguiente
    jmp     ver_mas

; Recupera el texto que contenía el EDITOR

fin_modo_conversacional:

    call    info                    ;Información de primera
    push    cs                      ;línea de la ventana del
                                    ;editor
    pop     es                      ;ES al segmento del
    assume  es:code                 ;programa.
    lea     si,area_ver_archivos    ;Ahora debe recuperar el
    lea    di,inicio_area_de_textos ;texto que estaba en el
```

```
    mov     cx,3900                 ;EDITOR.
    cld
    rep     movsw
    pop     inicio_pagina           ;Así mismo se recuperan
    pop     cursor_texto            ;la pos. del cursor de
    mov     attr_de_texto,17h       ;texto e inicio de
    ret                             ;página.
MODO_CONVERSACIONAL ENDP

;
;                                   CR_LF
;  Realiza un retorno del carro  o un avance de línea según
;  valor en AL.
;

CR_LF:
    push    cx                      ;CX y SI no deben
    push    si                      ;modificarse.
    cmp     al,13                   ;¿ INTRO ?
    je      do_cr                   ;Si, realiza un retorno
                                    ;del carro.
    cmp     al,10                   ;¿ LINE FEED ?
    je      do_lf                   ;Si, realiza un avance de
                                    ;línea
    mov     si,cursor_texto
    mov     byte ptr ds:[si],al     ;No, entonces almacene
    call    avz_cursor              ;caracter.
    jmp     fin_CR_LF
do_cr:
    call    inicio_de_linea         ;CR
    jmp     fin_CR_LF
do_lf:
    call    bajar_cursor            ;LF
fin_CR_LF:
    pop     si                      ;Recupera SI y CX
    pop     cx
    ret

;
;                              OPCIONES_DEL_TELEX
;        Chequea WHY, BELL, HERE_IS, MODO_CONVERSACIONAL.
;

OPCIONES_DEL_TELEX:
    cmp     al,11h                  ;¿ ^Q ? (¿Quién llama?)
    jne     ck_SOY                  ;No, chequear ^S
    call    sacar                   ;Si, enviar caracter de
    jmp     short TO                ;WHY y salir
ck_SOY:
    cmp     al,13h                  ;¿ ^S ? (Yo soy..)
    jne     ck_bell                 ;No, chequear ^G
```

```
    call    HERE_IS                 ;Si, enviar el indicativo
    jmp     short TO                ;y salir
ck_BELL:
    cmp     al,07                   ;¿ ^G ? (Sonido)
    jne     to1
    call    sacar                   ;Si, enviar caracter del
    call    parlante                ;BELL.
to:
    stc                             ;Al salir, CY=1 indicar
    ret                             ;que se realizó una de
to1:                                ;las opciones.
    clc                             ;CY=0 indica lo contrario
    ret

;
;   GRABAR_MENSAJE
;     Graba algún mensaje que se haya enviado o recibido
;

GRABAR_MENSAJE      PROC
    push    ax                      ;Se guardan todos los
    push    bx                      ;registros usados por
    push    cx                      ;esta subroutina
    push    dx
    push    si
    push    di
    push    ds
    push    es

    mov     ax,cs
    mov     ds,ax
    mov     es,ax

    mov     ax,3524h                ;Toma vector de Errores
                                    ;Críticos de Disco
    int     21h                     ;(INT 24h)
    push    bx                      ;Guarda vector en la pila
    push    es

    and     cs:bits_tlx,011b        ;Anula bit de grabado

; Sustituye la interrupción de errores del disco.

    mov     dx,offset io_error      ;Nueva INT 24h
    mov     ax,2524h
    int     21h


;  Ahora  trata  de  abrir el archivo para determinar  si  ya
;  existe.  Si es  así, entonces lo cierra  e  incrementa  el
;  nombre de archivos recibidos

    mov     dx,puntero_de_telex
```

```
    sub     dx,offset area_de_telex
    cmp     dx,30                   ;¿ Existen más de 30
                                    ;caracteres recibidos ?
    jbe     dos_error               ;No, entonces salir

abrir_archivo_tlx:
    lea     dx,refile               ;Cadena ASCIIZ contiene
                                    ;el nombre del mensaje
                                    ;recibido.
    mov     ax,3D00h                ;Función del DOS para
    int     21h                     ;abrir un archivo.
    jc      error_en_apertura       ;Si existe un error de
                                    ;apertura, salta.
    mov     bx,ax                   ;Manipulador del archivo
    mov     ah,3eh                  ;Cierra este archivo
    int     21h
    call    inc_nombre_de_archivo   ;Trata con el siguiente
    jmp     abrir_archivo_tlx       ;nombre de archivo.
error_en_apertura:
    cmp     ax,2                    ;¿Fue un error de archivo
                                    ;no encontrado ?
    je      crear_archivo           ;Si, crear el archivo
    call    parlante                ;No, salir
    jmp     dos_error

; Ahora crea el archivo, graba el contenido del mensaje
; recibido y cierra el archivo.

crear_archivo:
    lea     dx,refile               ;DX apunta al nombre del
                                    ;archivo.
    mov     cx,0020h                ;Atributo para el archivo
    mov     ah,3Ch                  ;Función para crear
    int     21h                     ;archivo para escritura.
    jnc     guardar_archivo
    call    parlante
    jmp     dos_error               ;Retorna bajo cualquier
guardar_archivo:                    ;tipo de error.
    mov     bx,ax                   ;Manipulador del archivo
                                    ;hacia BX.
    lea     dx,area_de_telex        ;Dirección desde donde se
    mov     cx,puntero_de_telex     ;grabará el texto.
    sub     cx,dx                   ;Calcula el número de
                                    ;bytes a grabar.
    mov     ah,40h                  ;Función del DOS para
                                    ;escribir en el fichero.
    int     21h                     ;Graba el mensaje.
cierra_archivo:
    mov     ah,3Eh                  ;Función del DOS para
    int     21h                     ;cerrar el archivo.
    call    inc_nombre_de_archivo   ;Siguiente nombre de
                                    ;archivo disponible
dos_error:
    lea     ax,area_de_telex
    mov     puntero_de_telex,ax
```

```
    mov     puntero_anterior,ax
    pop     ds                          ;Toma desde la pila el
    pop     dx                          ;vector de la INT 24h
    mov     ax,2524h                    ;Actualiza el vector de
    int     21h                         ;error.

    pop     es                          ;Recupera los registros
    pop     ds
    pop     di
    pop     si
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    ret                                 ;Finalmente sale
GRABAR_MENSAJE      ENDP


;
;   INC_NOMBRE_DE_ARCHIVO
;   Incrementa el nombre de archivo que contendrá el mensaje
;   recibido.
;


INC_NOMBRE_DE_ARCHIVO:
    lea     bx,archivo_de_telex+7   ;Apunta BX al final del
inc_sgte_caracter:                  ;nombre.
    inc     byte ptr DS:[BX]        ;Incrementa caracter.
    cmp     byte ptr DS:[BX],'9'    ;Chequea acarreo.
    jle     fin_inc                 ;Si no existe, sale.
    mov     byte ptr DS:[BX],'0'    ;De otro modo, fija a 0
                                    ;este dígito.
    dec     bx                      ;Retrocede al siguiente
                                    ;dígito
cmp bx,offset archivo_de_telex+3    ;¿ 4 dígitos chequeados ?
    jle     fin_inc                 ;Si, salir
    jmp     inc_sgte_caracter       ;Caso contrario, repite
fin_inc:                            ;el proceso.
    call    actualizar_directorio_tlx
    ret


;
;   RS232_IO
;   Subroutina para inicializar el puerto de Comunicación.
;


RS232_IO:
    mov     dx,puerto_serial        ;COM1 o COM2
    or      dx,dx                   ;¿ Existe algún puerto
                                    ;serial instalado ?
    jnz     inicalizar_rs232        ;Si, configurarlo
    ret                             ;Caso contrario, salir
```

```
inicalizar_rs232:

; Inicializa el puerto de comunicación para 50 baudios, 5
; bits de datos, 1 bit de parada, y no paridad.

    add     dx,3                    ;Apunta al REGISTRO DE
                                    ;CONTROL DE LINEA del
                                    ;8250 (3FB-2FB).
    in      al,dx                   ;Toma el estado de la
    or      al,80h                  ;línea.
    out     dx,al                   ;Fija DLAB=1.
    sub     dx,2                    ;Apunta DX al registro
    lea     si,bauds                ;divisor MSB (3F9-2F9).
    mov     al,cs:[si]+1            ;Coge valor MSB.
    out     dx,al                   ;Fija valor MSB en el
    dec     dx                      ;8250.
    mov     al,cs:[si]              ;Realiza lo mismo con el
    out     dx,al                   ;valor LSB (3F8-2F8).
    add     dx,3                    ;De nuevo hacia el RCL
                                    ;del 8250 (3FB-2FB).
    mov     al,bits                 ;Formato 5N1 y DLAB=0
    out     dx,al
    dec     dx                      ;DX hacia el Registro de
    dec     dx                      ;Habilitación de
    mov     al,0                    ;Interrupciones (3F9-2F9)
    out     dx,al                   ;Interrupciones anuladas.
    ret

;
;  IOSET: Fija los vectores de interrupción 1Bh, 23h, y
;         24h a manipuladores internos.
;  IORESET: Recupera los valores iniciales de los vectores
;

IOSET       PROC    NEAR

; Guarda el Area de Transferencia del Disco presente.

guardar_dta:
    mov     ah,2fh                  ;Función para obtener la
    int     21h                     ;dirección del DTA activo
    mov     old_dta_segment,es      ;Guarda DTA.
    mov     old_dta_offset,bx

; Fija nuevo DTA (Disk Transfer Area).

    lea     dx,dta
    mov     ah,1ah
    int     21h

; Guarda el Prefijo deI Segmento del Programa presente. PSP
; (Program Segment Prefix)
```

```
    mov     ah,51h                  ;Coge el segmento del PSP.
    int     21h
    mov     antiguo_psp,bx          ;y lo guarda

; Fija el PSP del Editor.

    mov     ah,50h                  ;Activa PSP del Editor
    push    cs
    pop     bx
    int     21h

; Guarda  y  modifica los vectores de interrupción 1Bh, 23h y
; 24h a manipuladores internos.

    mov     ax,351Bh                ;Obtiene la dirección de
    int     21h                     ;INT 1Bh.
    mov     old1Bh_segment,es       ;Guarda su segmento
    mov     old1Bh_offset,bx        ;y su desplazamiento
    mov     ah,25h
    mov     dx,offset io_salida     ;Nueva INT 1Bh.
    int     21h

; Interrupción  23h: Dirección de Ruptura,  anula   cualquier
; acción de CTRL-BREAK o CTRL-C

    mov     ax,3523h                ;Dirección de INT 23h.
    int     21h
    mov     old23h_segment,es
    mov     old23h_offset,bx
    mov     ah,25h
    mov     dx,offset io_salida     ;Nueva INT 23h.
    int     21h

; Interrupción 24h: Manipuladora  de  Errores  Criticos. Para
; las Versiones 3.X.

    mov     ax,3524h                ;Dirección de INT 24h.
    int     21h
    mov     old24h_segment,es
    mov     old24h_offset,bx
    mov     ah,25h
    mov     dx,offset io_error      ;Nueva INT 24h.
    int     21h
    ret
IOSET       ENDP

IORESET     PROC    NEAR

; Recupera PSP del programa interceptado.

    push    cs
    pop     es
    assume  es:nothing
    mov     ah,50h                  ;Recupera PSP
    mov     bx,antiguo_psp
```

```
        int     21h

;  Recupera DTA del programa interceptado o el usado  por  el
;  DOS.

        assume  ds:nothing
        mov     ds,old_dta_segment      ;Segmento y offset del
        mov     dx,old_dta_offset       ;DTA interceptado.
        mov     ah,1ah                  ;Recupera DTA
        int     21h

        mov     ax,2524h                ;Errores Criticos.
        mov     ds,old24h_segment
        mov     dx,old24h_offset
        int     21h

        mov     ax,2523h                ;Dirección de Ruptura.
        mov     ds,old23h_segment
        mov     dx,old23h_offset
        int     21h

        mov     ax,251Bh
        mov     ds,old1Bh_segment
        mov     dx,old1Bh_offset
        int     21h
        ret
IORESET ENDP


;
;  Espacio de datos usado por el programa  cuando permanece
;  residente en memoria.
;

        db      '       Env/rec. Mensaje: '
pc                      = $
area_de_telex           = pc            ;Area de mensajes
                                        ;recibidos.
pc                      = pc + 8000     ;8000 bytes reservados.
area_para_el_video      = pc            ;Area para reservar
                                        ;memoria de Video.
pc                      = pc+25*80*2    ;25 Lineas x 80 Columnas
                                        ;con Atributos.
inicio_area_de_textos   = pc            ;Area para almacenar
                                        ;TEXTOS.
pc                      = pc + 7800     ;100 Lineas x 78 Columnas
fin_area_de_textos      = pc
pc                      = pc + 78
current_dir             = pc            ;Directorio.
pc                      = pc + 68
dir_trabajo             = pc
pc                      = pc + 66
nombre_de_archivo_actual = pc           ;Nombre de archivo.
pc                      = pc + 13
```

```
fin_instalacion:
    mov     ah,9
    int     21h
    call    parlante
    ret

; Determina qué versión del DOS está corriendo.

no_instalado:
    mov     ah,30h
    int     21h
    lea     dx,dos_err
    cmp     al,3
    jb      fin_instalacion

; Guarda la dirección de la bandera interna del DOS (INDOS).

    mov     ah,34h
    int     21h
    mov     segmento_dos,es
    mov     indos,bx

; Guarda la dirección de la bandera de error crítico.

    mov     ax,3E80h
    mov     cx,2000h
    mov     di,bx
init2:
    repne   scasw
    jcxz    init3
    cmp     byte ptr es:[di+5],0BCh
    je      found
    jmp     short init2
init3:
    mov     cx,2000h
    inc     bx
    mov     di,bx
init4:
    repne   scasw
    jcxz    no_encontrada
    cmp     byte ptr es:[di+5],0BCh
    je      found
    jmp     short init4
no_encontrada:
    mov     ah,9
    lea     dx,errmsg2
    int     21h
    ret
found:      mov     ax,es:[di]
    mov     errflag,ax

; Determina la dirección del controlador del CRT.

    mov     ax,bios_seg             ;Apunta ES al área del
    mov     es,ax                   ;BIOS.
```

```
        assume  es:bios_seg

        mov     di,63h
        mov     dx,es:[di]              ;Toma la dirección base
        mov     direc_6845,dx           ;del CRT y la almacena.

; Determina qué tipo de adaptador de video está instalado.

        mov     adaptador,0             ;Asume que existe una MDA
        lea     si,attr_para_monoc      ;y un monitor monoc.
        mov     ah,12h                  ;Ve si una EGA está
        mov     bl,10h                  ;presente.
        int     10h
        cmp     bl,10h                  ;¿BL retorno con su mismo
                                        ;valor (10h) ?
        je      no_EGA                  ;Si, entonces no existe
                                        ;una EGA en el sistema
        test    estado_EGA,8            ;¿ Está la EGA activada ?
        jnz     no_EGA                  ;No, entonces chequear
                                        ;existencia de CGA o MDA.

; Existe en el sistema una tarjeta EGA

        mov     adaptador,2             ;Fija variable ADAPTADOR
                                        ;como EGA.
        lea     si,attr_para_monoc      ;Asume monitor monoc.
        or      bh,bh                   ;¿Está la EGA conectada a
                                        ;un monitor monoc. ?
        jne     fijar_param             ;Si, entonces fije
                                        ;parámetros.
        lea     si,attr_para_color      ;No, entonces apunte SI a
                                        ;los parámetros de color.
        jmp     short fijar_param

; Determina si el adaptador de video activo es un CGA o un MDA.

no_EGA: test    direc_6845,40h          ;¿ Existe un adaptador
                                        ;monocromático ?
        jz      fijar_param             ;Si, entonces fijar
                                        ;parámetros.
        mov     adaptador,1             ;No, fija ADAPTADOR como
                                        ;CGA.
        lea     si,attr_para_color      ;y apunta SI a los
                                        ;parámetros de color.

; Fija los parámetros de video para Color o Monocromático.

fijar_param:
        push    cs                      ;Apunta ES al segmento
        pop     es                      ;del programa.
        assume  es:code
        lea     di,sgmnto_de_video      ;DI destino
        mov     cx,8                    ;4 palabras a mover.
        cld                             ;SI y DI serán
                                        ;autoincrementados .
```

```
    rep     movsw                   ;Transfiere los valores.

;
;   Se modifican los vectores de interrupción para  que sean
;   interceptados por TELEX.
;

    mov     ax,3505h                ;Impresión de pantalla
    int     21h
    mov     old5h,bx
    mov     old5h[2],es
    mov     ah,25h
    lea     dx,int05
    int     21h

    mov     ax,3508h                ;Reloj principal
    int     21h
    mov     old8h,bx
    mov     old8h[2],es
    mov     ah,25h
    lea     dx,timer
    int     21h

    mov     ax,3509h                ;Teclado
    int     21h
    mov     old9h,bx
    mov     old9h[2],es
    mov     ah,25h
    lea     dx,teclado
    int     21h

    mov     ax,3510h                ;Video (BIOS)
    int     21h
    mov     old10h,bx
    mov     old10h[2],es
    mov     ah,25h
    lea     dx,video
    int     21h

    mov     ax,3513h                ;Disco (BIOS)
    int     21h
    mov     old13h,bx
    mov     old13h[2],es
    mov     ah,25h
    lea     dx,bios_disco
    int     21h

    mov     ax,351Ch                ;Tick del reloj.
    int     21h
    mov     old1Ch,bx
    mov     old1Ch[2],es
    mov     ah,25h
    lea     dx,intic
```

```
    int     21h

    mov     ax,3528h                ;Interrupción de  retorno
    int     21h                     ;de proceso del DOS.
    mov     old28h,bx
    mov     old28h[2],es
    mov     ah,25h
    lea     dx,backproc
    int     21h

    push    cs
    pop     es
    assume  es:code
    cld
    call    borrar_area_txt         ;Limpia el área de textos
    lea     ax,area_de_telex
    mov     puntero_de_telex,ax

; Ahora que casi todo está instalado, muestra el mensaje:
;   Telex en una PC/XT/AT 1.0
;   ESPOL-GYE 1989. Por Freddy Pinto Carpio
;   Secuencia de entrada: ALT-SHIFT izq.

    lea     dx,version
    mov     ah,09
    int     21h

; Chequea si es posible instalar TELEX en COM1 o en COM2.

    mov     bx,bios_seg             ;Fija ES al segmento del BIOS
    mov     es,bx
    assume  es:bios_seg
    mov     si,0                    ;COM1
    mov     al,1
sgte_puerto:
    mov     dx,rs232_base[si]       ;¿ Existe puerto serial ?
    or      dx,dx
    jne     puerto_encontrado       ;Si, mostrar mensaje.
    add     si,2                    ;COM2.
    inc     al
    cmp     si,4
    jne     sgte_puerto
    lea     si,errmsg4              ;Si no hay COM1 ni COM2,
    jmp     short mostrar_puerto    ;se muestra un mensaje de
puerto_encontrado:                      ;error.
    mov     puerto_serial,dx
    lea     si,puerto_de_telex
    or      al,30h                  ;Convierte 1 o 2 a valor
                                    ;ASCII.
    mov     byte ptr ds:[si+24],al  ;TELEX instalado en

mostrar_puerto:                     ;COM(1/2).
    mov     dx,si
    int     21h
    assume  es:nothing
```

```
; Ahora inicializa el puerto de comunicación para COM1 o COM2.

    call    rs232_io                ;Configura el puerto:
                                    ;50 bauds,5bits de datos,
                                    ;y 1.5 bits de parada.
    add     dx,3                    ;DX en el Registro de
                                    ;Control del Modem.
    mov     al,0                    ;DTR y RTS forzadas a
    out     dx,al                   ;nivel bajo.

; Fija el sendero donde se grabarán los mensajes recibidos

    call    actualizar_directorio_tlx

; Calcula la cantidad de memoria a reservar y termina pero
; permanece residente.

    mov     dx,(offset nueva_pila - offset code + 100 + 15)
    mov     cl,4
    shr     dx,cl
    mov     ax,3100h
    int     21h

initialize  endp

code        ends
    end     begin
```

APENDICE

MANUAL DEL USUARIO

El presente apéndice indica los pasos que deben seguirse para el correcto funcionamiento del programa que controla la interfase de télex.

Inserte el Diskette de arranque en la unidad de disco flexible y encienda su computador. (esto si el computador no tiene disco duro instalado).

Realizado el diagnóstico que el computador efectúa durante el encendido, se mostrará el prompt del DOS.

A:\>_ o C:\>_

El siguiente paso será crear un directorio llamado TELEX ya sea en la unidad A:, B: o en el mismo disco duro. Es importante hacer notar que si el computador solo tiene una unidad de diskette es necesario que siempre esté colocado un diskette en la unidad respectiva, puesto que al recibirse algún mensaje de télex, el programa de control tratará de grabarlo y si no existe diskette alguno en la unidad, puede perderse el mensaje con la llegada de otro.

Copie el archivo TELEX.COM en el directorio creado y digite en la posición del cursor lo siguiente:

C:\>**telex**

y presione la tecla **ENTER.**

El programa se cargará, y permanecerá residente en la memoria del computador, mostrando el siguiente mensaje:

Telex en una PC/XT/AT 1.0

ESPOL-GYE 1991. Por Freddy Pinto Carpio

Secuencia de entrada ALT-SHIFT izq

A partir de este momento es posible activar el programa simplemente presionando la secuencia ALT_SHIFT_IZQ ya sea desde la ejecución de algún programa de aplicación o simplemente desde el prompt del DOS. Una vez presionada esta secuencia de teclas se mostrará una ventana en la mitad de la pantalla esperando por el ingreso de algunos comandos suministrados para el control del editor de textos del télex.

ELABORACION DE UN TELEX.

El número del abonado a llamar debe colocarse al inicio de la primera línea del mensaje, seguida del signo '+'. Toda información que se escriba después del signo '+' en la primera línea del mensaje, no se envía. Sirve para referencia y comentarios. Esta información deberá estar separada del signo '+' por lo menos en un espacio en blanco.

A partir de la segunda línea se procede a elaborar el mensaje. Existen 100 líneas disponibles en el espacio del editor para la elaboración de un texto, aunque el envío de los mensajes no está restringido por la cantidad de líneas de texto a enviar. El final del mensaje debe ser indicado con 4 letras M's (mmmm). Esta secuencia informará a la central el fin de la comunicación; la misma que terminará automáticamente la llamada y la central de IETEL nos informará el tiempo que duró la comunicación.

Mientras se elabora un mensaje, la línea permanece desocupada, lo cual facilita la llegada de cualquier télex. Todo télex de llegada será recibido en letras mayúsculas. Debido a que el código BAUDOT está formado por de 5 bits, existen únicamente 32 combinaciones de caracteres que se puedan formar. Por lo tanto no existen los caracteres en minúsculas.

Todo télex que se prepare puede ser escrito en minúsculas, a fin de diferenciar claramente entre los mensajes recibidos de los enviados.

Una vez elaborado el mensaje, éste puede ser grabado como un archivo en el directorio TELEX, presionando la tecla F3 e indicando el nombre del archivo con el cual se grabará el mensaje; o si se quiere, es posible obtener una copia impresa del mismo.

ENVIO DE UN TELEX.

Para realizar el envío de un télex, se sigue el siguiente procedimiento.

Si el mensaje fue elaborado recientemente continuar con el paso siguiente. Caso contrario, elaborar el mensaje o leer el archivo de texto previamente grabado en el disco. Presionar la tecla F2 y especificar el nombre del archivo a leer. El archivo también puede editarse activando la ventana del DIRECTORIO y desde ésta seleccionar el archivo que contiene el mensaje a enviar.

Presionar la tecla DIAL F6, con lo cual el computador se comunica con la central de télex local.

Aparecerá en la primera línea de la ventana del editor de menú una letra 'C' y se encenderá el color rojo del led ubicado en la interfase de télex, lo que significa que la línea se ha invertido y conectado. Inmediátamente ingresará en números la fecha y hora actual seguidos del signo GA (Go Ahead) que significa 'siga adelante'.

Por ejemplo: 06.20. 17:30 GA

Lo cual significa Junio 20, 17 horas y 30 minutos (5.30 pm).

Si esto no sucede, significa que la línea de télex está ocupada, o que tiene fallas.

Las letras OCC indican que la línea está ocupada.

Las letras **DER** indican que la línea y/o el equipo del abonado al cual queremos llamar, estan averiados.

Si aparece GA, se presiona nuevamente la tecla **F6**, por segunda vez, para marcar el número de télex que se encuentra en la primera línea del mensaje. La central de télex nos envía de nuevo el número al cual estamos llamando junto con el indicativo del abonado. Esto sirve para verificar que el número al cual hemos marcado es el correcto.

Se observa en la ventana de envio/recepción el número marcado y a continuación la respuesta del otro equipo, que aparecerá de la siguiente forma por ejemplo:

43509+ ESPOL ED

En caso de ser una llamada local, se muestra el número y el indicativo del abonado del número de télex marcado. Cuando es una llamada internacional, responde la central de télex internacional y a continuación ingresa el número y el indicativo del número marcado.

Cuando se dificulta la comunicación con el número marcado aparecen la letras **NC**, indicando que no es posible realizar la comunicación.

Si aparecen las letras **NA** o **NP**, significa que existe algún error en el número que deseamos discar.

Una vez establecida la comunicación con el abonado deseado, podemos enviar la información que contiene el editor,

solo presionando la tecla de envío F7. El mensaje que se encuentra a partir de la segunda línea del editor empezará a ser enviado caracter por caracter a través de la línea de comunicación y cada caracter enviado es mostrado en la ventana de envío/recepción de mensajes.

Durante el envío del mensaje, es posible parar temporalmente su transmisión sin que esto corte la llamada. Para hacer esto se debe presionar la tecla STOP SEND F8 e inmediátamente la transmisión será interrumpida hasta que el usuario presione nuevamente la tecla de envío F7. Por el contrario, si se quiere parar el envío y también cortar la comunicación con la línea de télex, podemos presionar la tecla HANG UP F9 con lo cual el programa enviará las señales de pulso necesarias para que la línea de télex se invierta, y corte la comunicación.

Si se escribieron las 4 m's (mmmm) al final del texto, se cancelará automáticamente la llamada y se realizará la impresión del original enviado.

Cuando una llamada es cancelada mediante la tecla F9, la central no nos proporciona el tiempo que duró la comunicación; por lo tanto, para poder cerrar la llamada y obtener el tiempo de conexión, podemos presionar la secuencia CTRL-V para ingresar al modo Conversacional e inmediátamente digitamos o las 4 m's o también algunos puntos suspensivos (.....), con lo cual la llamada se cierra y obtenemos el tiempo que duró la comunicación. Esto se realiza cuando el final del mensaje no contiene las 4 M's. Es importante tener

presente de que el mensaje no debe contener las 4 M's como parte de su contenido puesto que si las 4 M's están por ejemplo al inicio del mensaje, en el monento de su transmisión la central interpretará esta secuencia como fin del mensaje y cortará la comunicación, quedando parte del mensaje sin enviar.

Veamos un pequeño ejemplo:

Llamada a ESPOL:

Télex: 43509

Indicativo: ESPOL ED

1) Elaboración del télex:

Linea

1: 43509+ Télex de prueba

2:

3: Saludos

4: Esta es una prueba de envio

5: Por favor devolver este mismo mensaje

6: para conocer resultados de la transmisión

7: mmmm

2) Secuencia de envio:

DIAL F6

La Central responde: 11.15 12:40 GA

DIAL F6 43509+

La Central responde: 43509 ESPOL ED

SEND F7 Saludos

(continuación del mensaje)

mmmm

La Central responde: (tiempo de conexión)

RECEPCION DE UN TELEX

Toda llamada entrante a más de ser señalizada acústicamente y de ingresar automáticamente èn la memoria del computador, se la puede observar en la ventana de recepción ubicada en la parte superior derecha de la pantalla. Todo télex que llega se almacena en la unidad disco del computador en forma de un archivo ASCII con el nombre TEXTnnnn y extensión TLX, donde nnnn puede ser un número entre 0000 y 9999. Al momento de la r' cepción no es necesaria la intervención de un operador, ni mpoco es necesario activar el programa con la secuencia T_SHIFT_IZQ. Es más, el usuario puede estar realizando ras tareas en el computador. Por ejemplo, puede estar abajando con una hoja de cálculos con el programa LOTUS y mismo tiempo puede estar observando la llegada del mensaje r la vantana de recepción sin que esto interrumpa su tarea.

)DO CONVERSACIONAL

forma más indicada para mantener una conversación a través la línea de télex independientemente de quién realizó la amada, es presionando simultáneamente la secuencia CTRL-V, spués de que el programa TELEX haya sido activado. Con esto e mostrará una nueva ventana que nos permitirá realizar intercambio de frases con el abonado. En estas circunstancias, todo lo que digitemos será inmediátamente

coexistir sin que se produzcan inconvenientes.

CONCLUSIONES Y RECOMENDACIONES

El presente trabajo ha demostrado la facilidad de emplear un computador personal como un excelente sistema para el envio y recepción de mensajes de télex sin necesidad de que el computador quede dedicado a atender solo a los servicios de la línea de télex.

Este sistema fue probado en algunos modelos de computadores IBM y compatibles; incluso con el sistema multiusuario NOVELL Versión 3.15, desde una estación de trabajo obteniéndose resultados positivos; y, como hemos visto, ha sido desarrollado tomando en cuenta muchas posibles fallas tratando de evitarlas de la mejor forma posible.

Como recomendaciones podemos citar las siguientes:

- Evitar dar ALT-CTRL-DEL durante la recepción de mensajes, aunque si está activada la ventana de TELEX, esta secuencia no afecta la operación normal del computador. Esta recomendación se da en el caso de que se esté ejecutando otra aplicación y a la vez se esté recibiendo información. Tampoco apague el computador durante estas condiciones.

- Tratar de no tener cargados a memoria demasiados programas residentes. Esto porque además de restar la memoria disponible para los programas de aplicación puede afectar con la operación normal del programa TELEX. No todos los programas residentes pueden coexistir al mismo tiempo en la memoria del computador.

- Siempre mantener una copia impresa tanto de los mensajes enviados como de los mensajes recibidos además de un respaldo a diskette de la información de los archivos de télex.

- Si bien es cierto, la línea corta la comunicación automáticamente después de 3 minutos de ausencia de envio o recepción de información es necesario asegurarse del fin de enlace y que la línea queda en su estado normal.

- No olvidar que la primera línea del editor de textos sirve para colocar el número del abonado a llamar y que debe finalizar con el signo '+'. Si el número es colocado sin este signo, al intentar realizar una llamada se mostrará un mensaje de error.

- Recuérdese que el editor de textos de TELEX tiene espacio para 100 lineas de 66 caracteres. Si se desea enviar un télex con más número de líneas se recomienda crear archivos adicionales y enviarlos uno a continuación de otro. Puesto que TELEX es un programa residente, se ha tratado de no usar demasiada memoria para el editor.

- Evitar ejecutar programas de diagnóstico al computador mientras TELEX esté cargado a memoria. Algunos programas de este tipo borran el contenido de la memoria o finalizan con un reset al computador, y puede darse el caso de que en ese momento esté ingresando un mensaje al computador.

- Si un puerto serial ha sido destinado para la comunicación con la interfase de télex, se recomienda no correr programas de comunicación que accesen a tal puerto. Estos programas pueden cambiar los parámetros del puerto (velocidad, bit de datos, paridad, etc). Aunque TELEX, siempre trata de que estos parámetros no cambien.

- Evitar enviar información al mismo tiempo que se está recibiendo. La línea de télex opera en el modo simplex y por lo tanto, la transmisión simultánea de ambos abonados hará que ambos reciban información errónea.

- No correr programas que no permiten el retorno al DOS, como por ejemplo programas de juegos, demostraciones, etc.

- Puesto que se supone que el computador debe estar encendido durante las 24 horas del día para atender el ingreso de mensajes a cualquier hora, se recomienda proteger al equipo con un buen regulador de voltaje y con un UPS para evitar pérdida de información durante ausencia de energia eléctrica.

El uso de los puertos de entrada/salida del computador representa una experiencia positiva y provechosa puesto que es posible controlar a través de estos desde sencillos

dispositivos digitales hasta elementos electromecánicos e incluso plantas industriales. Para lograr esto es muy importante conocer las características del sistema que se desea controlar, sus entradas, sus salidas, los niveles de voltaje para poder asociarlos con una adecuada interfase de control.

B I B L I O G R A F I A

1. THE IBM PERSONAL COMPUTER: FROM THE INSIDE OUT
   Sargent-Shoemaker: Capítulos 2 y 3

2. INSIDE THE IBM PC´S.
   BYTE: Volumen 12, número 12, 1987. Pág.173-180

3. INTRODUCCION AL TELEPROCESAMIENTO
   James Martin: Capítulo 6, Pág. 85-88

4. INTERCONECION DE PERIFERICOS A MICROPROCESADORES
   Mundo Electrónico: Victor Gilgales, Cap. 5 Pág. 100-108

5. SPEAKING THE RS-232 LANGUAGE.
   PC MAGAZINE, Julio 15, 1988 Pág. 374-375

6. THE ASYNCHRONOUS ADAPTER ANS RS-232.
   PC MAGAZINE, Enero 17, 1989 Pág. 275-276

7. PROGRAMMING THE 8250 UART.
   PC MAGAZINE, Octubre 13, 1987 Pág. 434-435

8. INSTANT ACCESS TO DIRECTORIES.
   PC MAGAZINE, Abril 14, 1987 Pág. 313-334