

12  
2ej.



**UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO**

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES  
" ARAGON "

**DISEÑO Y CONSTRUCCION DE UNA TARJETA DE EXPANSION  
O INTERFAZ PARA UNA PC-XT COMPATIBLE CON IBM,  
APLICABLE AL CONTROL DE UNA LECTORA OPTICA.**

TESIS CON  
FACULTAD DE INGENIERIA

**TESIS PROFESIONAL**

QUE PARA OBTENER EL TITULO DE:  
INGENIERO MECANICO ELECTRICISTA  
P R E S E N T A :  
JOSE LUIS LEGORRETA GARCIA

DIRECTOR DE TESIS:  
ING. RICARDO MARTINEZGARZA FERNANDEZ

MEXICO, D. F.

1992



Universidad Nacional  
Autónoma de México



## **UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

### **DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

## C O N T E N I D O

INTRODUCCION .....	i
CAPITULO I: LA LECTORA OPTICA Y LA PC-XT .....	1
CAPITULO II: ESTRUCTURA GENERAL DEL SISTEMA ELECTRONICO ..	5
CAPITULO III: PROGRAMA DE CONTROL DE LA INTERFAZ .....	30
CAPITULO IV: MANUAL DE USUARIO .....	67
CONCLUSIONES .....	71
APENDICE " A " .....	72
BIBLIOGRAFIA .....	73

## I N T R O D U C C I O N

En las últimas décadas se ha observado un desarrollo en el aspecto técnico y tecnológico a pasos agigantados en todo el mundo y fundamentalmente en las áreas de electrónica, telecomunicaciones y computación en donde se manifiestan plenamente estos avances que no dejan de sorprendernos. Es por esto que la U.N.A.M., al ser una de las Universidades más prestigiadas del País, lucha día a día por estar a la vanguardia en el ámbito de la tecnología, además de obtener el máximo provecho de sus equipos existentes por medio de interfaces de control, interconexión de nuevos sistemas y equipos, dispositivos, etc.

En este sentido, el trabajo presentado en esta tesis está encaminado a obtener el máximo provecho de las tres lectoras ópticas utilizadas por la U.N.A.M., al sustituir los sistemas actuales de control que son minicomputadoras HP-1000.

Las lectoras ópticas son utilizadas por la U.N.A.M. para leer información sobre exámenes de admisión a nivel bachillerato y licenciatura, para la reinscripción de alumnos en los niveles de bachillerato y licenciatura, así como para los registros de los exámenes extraordinarios en los niveles anteriormente citados.

También se lee información de otras Universidades del interior de la República e incluso algunas dependencias del Gobierno dentro del D.F. y área Metropolitana.

En general, se lee un promedio del 85% del total de la información que ingresa a la U.N.A.M. a través de la lectura óptica.

Además de las aplicaciones anteriores, la Dirección General de Servicios Médicos ocupa el servicio de lectura óptica para capturar la información de las encuestas de salud de los alumnos de nuevo ingreso a la U.N.A.M. en sus diferentes niveles. Las Facultades de Medicina y Odontología ocupan este servicio además para leer información de exámenes profesionales.

Como podrá observarse, cualquier información que se relacione con respuestas de opción múltiple, podrá manejarse a través de lectura óptica.

La U.N.A.M. cuenta en su nivel Bachillerato con nueve Preparatorias y cinco Colegios de Ciencias y Humanidades; en el nivel superior, además de todas las Facultades, Direcciones, Institutos y Dependencias, cuenta con cinco planteles repartidos en el área Metropolitana, (cuatro Escuelas Nacionales de Estudios Profesionales y una Facultad de Estudios Superiores).

En el periodo de Enero a Septiembre de 1991, se leyó un total de 1'521,577 hojas para la U.N.A.M., mientras que para Universidades foráneas se leyó un total de 41,836 hojas en este mismo periodo.

La U.N.A.M. cuenta con tres lectoras ópticas y dos minicomputadoras HP-1000 por lo que se hace necesario eliminar el actual sistema de control dado que cada lectora óptica debe ir conectada a una minicomputadora HP-1000, cuya vida útil ha concluido.

Las minicomputadoras HP-1000 utilizadas para el control de los equipos de lectura óptica son a la fecha, equipos obsoletos debido a que su arquitectura fué modificada para lograr el control, cuenta con una interfaz especial y por tanto el software también difiere del resto de los equipos HP-1000. Esto significa que tienen una circuitería muy especial que no es compatible con equipos de su clase, por lo que es difícil encontrar refacciones de dicho equipo en el País, además del alto costo de mantenimiento preventivo y correctivo.

Otra de las desventajas es que es un equipo grande y voluminoso en relación con las actuales PC-XT que, además de ser pequeñas en tamaño, ofrecen mucho más ventajas y mayor velocidad en el procesamiento de la información.

Este proyecto de tesis surgió de la necesidad de mantener en servicio las lectoras existentes, utilizando tecnología moderna de control y de bajo costo. Por tal motivo se decidió ocupar los recursos existentes en aquellas fechas y utilizar como sistema de control una microcomputadora PC-XT compatible con IBM, obteniendo con ello una mayor eficiencia.

## CAPITULO I. LA LECTORA OPTICA Y LA PC-XT

### A. LA LECTORA OPTICA

#### A.1 PANORAMA GENERAL DE LA LECTORA OPTICA

La lectora óptica modelo W201 está diseñada para trabajos pesados de lectura de información a altas velocidades. La velocidad se puede elegir en tres opciones que nos brinda la lectora y que pueden ser 80, 100 y 170 hojas por minuto.

Cuenta con una cámara electrónica para convertir imágenes ópticas, reflejadas de la superficie de las formas que pasan a través de la estación de lectura, en señales electrónicas. Esta información es procesada por circuitería electrónica integrada dentro de la lectora óptica y pasada a una minicomputadora de propósito general.

La minicomputadora puede ser programada para manipular los datos de alguna manera especial. A esta manipulación especial se le conoce como aplicación y consiste en recopilar la información obtenida de la lectora óptica en formatos establecidos previamente para posteriormente ser decodificados y obtener la información de cada una de las formas leídas.

Varios modelos de minicomputadoras Hewlett-Packard son usadas como elemento de control para la lectora óptica.

#### A.2 HOJAS DE LECTURA OPTICA

Las hojas de lectura óptica tienen en la columna externa izquierda unas marcas de señalización de color oscuro las cuales se utilizan para identificar los renglones en los que se presenta la información y se conocen como márgenes de registro. Estos márgenes de registro son precisamente los que detecta la lectora óptica para ordenar que se lleve a cabo el barrido de lectura en el renglón en el que se encuentra la marca de registro.

Las hojas de lectura óptica son tamaño carta y contienen 61 renglones por cada lado. Los renglones contienen una secuencia de óvalos relacionados cada uno de ellos a un número y una serie de letras determinadas; estos se marcan o rellenan con lápiz del número dos o dos y medio para que la lectora óptica detecte esta marca y la reconozca como información.

El número del lápiz mencionado es requisito indispensable para el llenado de los óvalos debido a que el grafito es más suave y deja una marca considerablemente más oscura y a su vez elimina la posibilidad de que la lectora no reconozca dichas marcas.

El formato típico de una hoja de lectura óptica se muestra en la figura 1.1.

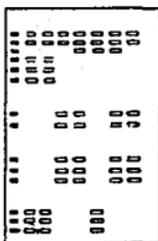


Figura 1.1 Formato típico de una hoja de lectura óptica.

Los formatos de las hojas varían de acuerdo al tipo de aplicación de éstas y las aplicaciones principales utilizadas por la U.N.A.M. se describen a continuación:

- \* Exámenes de admisión a nivel bachillerato y licenciatura
- \* Exámenes extraordinarios
- \* Registro de reinscripción a los semestres posteriores al primero en los niveles de bachillerato y licenciatura
- \* Exámenes para el diagnóstico del estado de salud física de las personas
- \* Exámenes para el diagnóstico del estado de salud mental de las personas
- \* Exámenes profesionales para la Facultad de Medicina

### A.3 ESTACION DE LECTURA

La estación de lectura es una lectora reflectiva de información. Las hojas son leídas en la estación de lectura por ambos lados en un sencillo paso en el que primeramente se lee la información del frente y posteriormente el reverso de la hoja.

La estación de lectura cuenta con dos lámparas que iluminan las hojas y la luz reflejada de éstas es dirigida al lente de la cámara por medio de un espejo. Dentro de los lentes, la luz pasa a través de un filtro infrarrojo y ésta se dirige a un dispositivo acoplado por carga (CCD).

El CCD es el dispositivo encargado de convertir las señales ópticas en señales eléctricas.

Las señales eléctricas son transferidas a un registro de corrimiento y este a su vez envía un stream de bits seriales a la lógica del sistema. Los datos de los procesos lógicos de lectura se almacenan en RAM.

La fig. 1.2 muestra la estación de lectura y la cámara correspondientes a la lectora óptica W201.

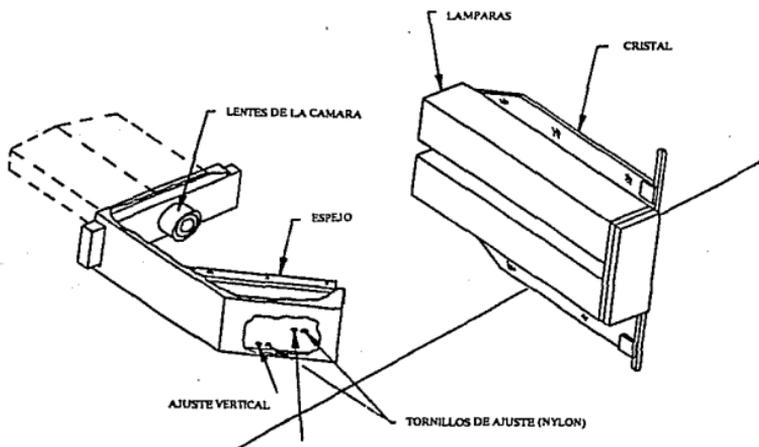


Figura 1.2 Estación de lectura y cámara de la W201

#### **A.4 SENSIBILIDAD DE LA LECTORA OPTICA**

La lectora óptica cuenta con un grupo de micro-interruptores que nos permiten variar la sensibilidad de la lectura en condiciones normales de operación. Esto es muy importante debido a que en ocasiones se presentan grupos de hojas con manchas y borrones, por ejemplo, y a través de estos micro-interruptores el operador incrementa el peso de la sensibilidad para que la lectora óptica ignore dichas manchas y borrones. De otra forma, si un grupo de hojas ha sido llenado con lápiz de grafito del tipo duro, el operador puede decrementar el peso de la sensibilidad con objeto de incrementar la habilidad para discriminar entre varios niveles.

#### **A.5 CIRCUITOS DETECTORES DE LA LECTORA OPTICA**

Cuando una hoja entra a la estación de lectura, un circuito de chequeo de lectura actúa para detectar algún material suelto sobre el cristal, espejo o lentes, que será leído como una falsa marca. Una condición de error de lectura causa que la lectora óptica se detenga y que envíe un mensaje al sistema de control.

El periodo de lectura inicia desde el momento en el que la hoja llega a la estación de lectura. Determinados circuitos se activan a la vez para asegurar la lectura de los datos correctos o para detectar condiciones de error que afectarán la validez de los datos leídos.

La lectora óptica tiene la facilidad de detectar los siguientes estados de error:

- \* Desviación
- \* Hojas múltiples
- \* Atascamiento

#### **DETECTOR DE DESVIACION**

Este circuito se encarga de hacer una revisión para comprobar que la hoja esté correctamente alineada respecto al borde de la guía de la lectora. El parámetro utilizado para definir si está o no alineada la hoja es precisamente el margen de registro. La lectora óptica contiene un grupo de tres micro-interruptores que de acuerdo a la configuración de estos, permiten la lectura de hojas ligeramente mal arregladas o con márgenes de registro ligeramente dañados.

## **DETECTOR MULTI-HOJA**

Este circuito esta compuesto de un led que emite luz sobre un lado de la hoja, y una fotocelda, el conjunto led-fotocelda detecta la cantidad de luz transmitida a través de la hoja. El circuito tiene una salida de lectura digital que permite el ajuste para condiciones de una-hoja o multi-hoja durante la calibración. Durante la operación, si se presenta la condición multi-hojas, el motor de transporte será detenido.

## **DETECTORES DE ATASCAMIENTO**

Estos circuitos contienen transmisores y detectores de luz similar a los detectores multi-hoja y monitorean el paso de la hoja a lo largo de la trayectoria de transporte. La longitud de la hoja es medida cuando entra al mecanismo de transporte y este dato es alimentado al primero de varios registros de corrimiento conectados en serie, formando una larga línea de espera con detectores en puntos apropiados. La distribución de estos registros de corrimiento permite a la lectora óptica mantener varias hojas sobre la trayectoria de transporte al mismo tiempo. Cuando uno de estos registros no recibe la información del valor correcto de la longitud de la hoja original con respecto a la hoja detectada, inmediatamente la lectora envía un comando para detener el motor.

En el panel frontal de la lectora óptica se muestra la trayectoria de transporte y se encenderá un led mostrando en que parte ocurrió el atascamiento.

### **A.6 PALABRAS DE ESTADO DE LA LECTORA OPTICA**

El CPU es informado del estado de la lectora óptica por medio de tres palabras transmitidas por la lectora. En resumen, el contenido de las tres palabras de estado son:

**PALABRA 1:** Errores de hardware de la lectora y condiciones de no-lista.

**PALABRA 2:** Dato de hoja desviada.

**PALABRA 3:** Condición de hoja volteada (la información está siendo leída).

Como puede observarse, cualquier evaluación que se elabore para ser contestada en base a varias alternativas de respuesta, se puede llevar a cabo mediante el proceso de lectura óptica.

## B. LA PC-XT

### B.1 PANORAMA GENERAL DE LA PC-XT

La PC-XT es una computadora del tipo personal constituida por un CPU, memoria y circuitería de entrada/salida. Estas tres partes se interconectan en conjunto a través del bus de control, bus de datos y bus de direcciones.

La secuencia de instrucciones o programa de control de la microcomputadora es almacenado en forma binaria en localidades sucesivas de memoria.

El CPU toma una instrucción de memoria, la decodifica para determinar que acción será realizada y la ejecuta.

La microcomputadora PC-XT cuenta además con buses de expansión que le permiten la comunicación al mundo exterior a través de diferentes dispositivos conocidos como equipos periféricos.

### B.2 BUS DE EXPANSION DE LA PC-XT

El bus de expansión de la PC-XT compatible con IBM consta de 62 conexiones mostradas en la figura 1.3. Muchas de las conexiones son usadas para el acceso directo de memoria y para el manejo de las interrupciones. También contempla líneas de datos, líneas de control y líneas de direcciones.

La polarización para los equipos periféricos es proporcionada en las líneas B1 y B31 (tierras) y en las líneas B3 y B29 (+5V). Todas las líneas del bus de expansión son compatibles con niveles TTL.

El bus de expansión contiene los siguientes elementos:

- \* 1 bus de datos bidireccional de 8 bits
- \* 20 líneas de dirección
- \* 6 niveles de interrupción
- \* 3 canales para líneas de control DMA
- \* 1 canal para pruebas de línea
- \* Líneas de control para lectura de memoria y lectura o escritura de entrada/salida
- \* energía y tierra para los adaptadores

NOMBRE DE LA SEÑAL

BUS DE EXPANSION  
-----

NOMBRE DE LA SEÑAL

	GND	B1	A1	-I/O CH CK
+RESET	DRV	B2	A2	+D7
	+5V	B3	A3	+D6
	+IRQ2	B4	A4	+D5
	-5V	B5	A5	+D4
	+DRQ2	B6	A6	+D3
	-12V	B7	A7	+D2
-CARD	SLCTD	B8	A8	+D1
	+12V	B9	A9	+D0
	GND	B10	A10	+I/O CH RDY
	-MEMW	B11	A11	+AEN
	-MEMR	B12	A12	+A19
	-IOW	B13	A13	+A18
	-IOR	B14	A14	+A17
	-DACK3	B15	A15	+A16
	+DRQ3	B16	A16	+A15
	-DACK1	B17	A17	+A14
	+DRQ1	B18	A18	+A13
	-DACK0	B19	A19	+A12
	CLOCK	B20	A20	+A11
	+IRQ7	B21	A21	+A10
	+IRQ6	B22	A22	+A9
	+IRQ5	B23	A23	+A8
	+IRQ4	B24	A24	+A7
	+IRQ3	B25	A25	+A6
	-DACK2	B26	A26	+A5
	+T/C	B27	A27	+A4
	+ALE	B28	A28	+A3
	+5V	B29	A29	+A2
	+OSC	B30	A30	+A1
	GND	B31	A31	+A0

FIGURA 1.3

Bus de entrada/salida de una microcomputadora PC-XT compatible con IBM. Todas las señales tienen niveles de voltaje TTL (0 a 5 V).

### B.3 DESCRIPCION DE LAS SEÑALES DEL BUS DE EXPANSION

A continuación se dará una descripción de las señales involucradas en el bus de expansión.

#### +{AO-A19}

Son las direcciones de los bits 0 al 19 que se utilizan para direccionar la memoria y los dispositivos de entrada/salida del sistema. Estas líneas pueden ser generadas por el procesador o el controlador DMA (Direct Memory Access).

#### +ALE

(Address Latch Enable). Esta señal habilita los acondicionadores de las direcciones de salida del 8088.

#### CLOCK

Reloj del sistema con una frecuencia de 4.77 MHz y un ciclo de trabajo del 33 %, producida por el generador de reloj 8284 (la frecuencia de 14.318 MHz del cristal se divide entre tres).

#### +AEN

(Address Enable). Inhibe al controlador del bus 8288 y a los acondicionadores de las direcciones para permitir que el DMA asuma el control del bus.

#### +{D0-D7}

Líneas de datos que constituyen el bus bidireccional de 8 bits para transferencia de datos, comandos e información.

#### -{DACK 0-3}

(DMA Acknowledge Channels). Estos canales de reconocimiento del DMA activos bajos, notifican a dispositivos periféricos individuales cuando se ha otorgado un servicio del DMA.

#### +{DRQ 1-3}

(DMA Request Channels). Un dispositivo periférico puede obtener servicio del DMA a través de estos canales de petición de entrada.

+T/C

Señal de salida que representa la terminación de un servicio del DMA.

-I/O CH CK

(I/O Channel Check). Señal generada por una tarjeta externa en el bus de expansión que informa al CPU de una falla catastrófica por medio de la circuitería del NMI (No Mascarable Interruption).

+I/O CH RDY

(I/O Channel Ready). Señal de "handshake" (protocolo) para una tarjeta externa en el bus de expansión empleada cuando se insertan estados de espera a los ciclos de reloj del CPU o un DMA.

-IOR

(I/O Read). El CPU activa en bajo esta señal para realizar una lectura hacia un dispositivo periférico del sistema.

-IOW

(I/O Write). El CPU activa en bajo esta señal para realizar una escritura hacia un dispositivo periférico del sistema.

+ [IRQ 2-7]

(Interrupt Request Channels). Estas señales de entrada son canales de petición de interrupción pertenecientes al 8259; los niveles de prioridad de los mismos son establecidos por el diseñador del sistema.

-MEMR

(Memory Read). Señal que el CPU habilita en bajo cuando realiza una lectura a memoria.

-MEMW

(Memory Write). Señal que el CPU habilita en bajo cuando realiza una escritura a memoria.

**+OSC**

Señal cuya frecuencia es equivalente a la frecuencia del cristal de 14.318 MHz, usada exclusivamente por la circuitería de video.

**+RESET**

Señal para reestablecer o inicializar el sistema lógico en el encendido de la máquina o durante bajas de voltaje.

Los voltajes que se enlistan a continuación los suministra directamente el bus de expansión:

+ 5 V	+ 5%
- 5 V	+10%
+12 V	+ 5%
-12 V	+10%
GND	

NOTA: Las señales antepuestas con el signo "+" son consideradas activas altas, mientras que las señales con signo "-" son consideradas como activas bajas.

El mapa de direccionamiento de líneas de entrada/salida de una PC-XT compatible con IBM se muestra en la siguiente página:

**DIRECCION**  
**(Hexadecimal)**

**APLICACION**

000-00F	Controlador de acceso directo a memoria (DMA) 8237A-5.
020-021	Control de interrupciones programables 8259A.
040-043	Timer 8253-5.
060-063	Controlador de interfaz programable paralelo 8255A-5.
080-083	Registros de página DMA.
0AX	Registros de interrupción no mascarables.
0CX	Reservado.
0EX	Reservado.
200-20F	Control de juegos.
210-217	Unidad de expansión.
220-24F	Reservado.
278-27F	Reservado.
2F0-2F7	Reservado.
2F8-2FF	Comunicación asíncrona (secundaria).
300-31F	<b>Tarjeta prototipo.</b>
320-32F	Disco duro.
378-37F	Impresora.
380-38C *	Comunicación SDLC.
380-389 *	Comunicación síncrona binaria (secundaria).
3A0-3A9	Comunicación síncrona binaria (primaria).
3B0-3BF	Monitor monocromático e impresora.
3C0-3CF	Reservado.
3D0-3DF	Gráficas a color.
3E0-3E7	Reservado.
3F0-3F7	Disco.
3F8-3FF	Comunicación asíncrona (primaria).

\* No podrán utilizarse estas direcciones para ambas comunicaciones, solo una a la vez.

TABLA 1-1. Espacio de direccionamiento de I/O de una PC-XT compatible con IBM.

#### B.4 SEÑALES INVOLUCRADAS EN EL CICLO I/O DE LA PC-XT

La figura 1.4 representa las señales involucradas en el ciclo de I/O de la PC-XT, mostrándolo por palabra, utilizando dos ciclos de direcciones sucesivos, es decir, para completar la palabra de I/O primeramente se trae el byte menos significativo (LSB) que es la dirección baja y posteriormente el byte más significativo (MSB) que es la dirección más significativa llevada a cabo por el microprocesador 8088.

Las líneas de direcciones A0 a A9 determinan cuál tarjeta es seleccionada por el bus. El intervalo más pequeño entre los dos ciclos indica qué dato es considerado como la dirección alta, pero no todos los circuitos integrados periféricos pueden manipular dos accesos al mismo tiempo.

La señal de reloj es marcada con un espacio a razón de 2 a 1 con un tiempo bajo de 140 ns. y un tiempo alto de 70 ns. Un simple byte de acceso de I/O requiere 5 ciclos de reloj completos sobre el primer ciclo T4. Es por eso que el byte de I/O puede completarse con un mínimo de 1.02 microsegundos a 4.77 MHz. "Turbo" de la PC-XT compatible con IBM puede tener un tiempo mayor.

Se puede llevar a cabo la extensión de los ciclos de lectura/escritura de la PC-XT usando I/O Channel Ready (I/O CH RDY). Cuando una tarjeta lenta es seleccionada ésta activa al buffer en tercer estado sujetando la línea de I/O CH RDY y la PC sensa este nivel en el ciclo T2 como se muestra en la figura 1.4.

Ahora bien, si I/O CH RDY es activa baja (el dispositivo aún no está listo), la PC automáticamente inserta un estado de wait dentro del ciclo de I/O y la línea es sensada otra vez. La PC suma otro ciclo de reloj extra (TW1) en el primer ciclo de I/O, después I/O CH RDY es activa baja al final del ciclo T2.

Hay que hacer notar que cuando la tarjeta del drive está en tercer estado por la línea I/O CH RDY ésta será activada en alto para el estado ready.

Retrasando el ciclo de I/O por algunos microsegundos causará problemas con el refresco de memoria del sistema. El diseño proporcionado no requerirá más que unos ciclos extras de reloj para operar apropiadamente.

La PC-XT compatible con IBM tiene restricciones más incómodas. Aunque el microprocesador 8088 puede utilizar todas las 15 líneas de direcciones para acceder 64 Kbytes de espacio de I/O, sólo las

lineas de direcciones (A0-A9) son actualmente decodificadas para I/O resultando un espacio restringido de 1 Kbyte.

La PC-XT compatible con IBM asigna localidades estandar de I/O a muchos dispositivos de esta región de 1 Kbyte y los asigna como se muestra en la tabla 1-1.

A primera vista podría parecer que hay espacio en la región "tarjetas prototipo", cuyas direcciones corresponden a la 300-31F hexadecimal. Sin embargo, esta región es utilizada en la mayoría de los casos por otras tarjetas que manipulan periféricos.

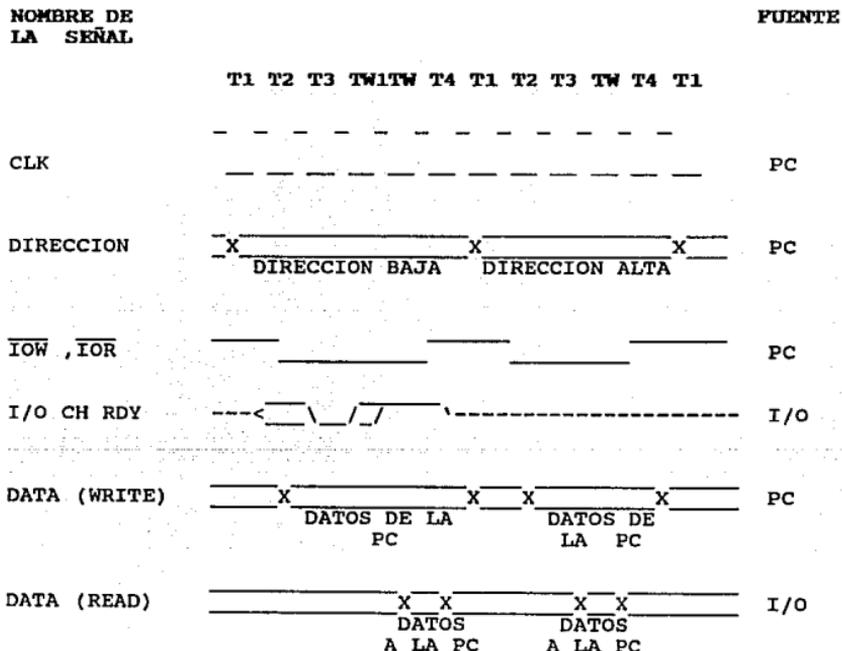


FIGURA 1.4 Diagrama de tiempos del ciclo de I/O por palabra de una PC-XT compatible con IBM. El intervalo de CLK es de 210 ns.

NOTA: El ciclo de una palabra es llevado a cabo por dos bytes del ciclo de I/O y el I/O CH RDY es utilizado para sumar un ciclo extra de reloj en el primer ciclo de I/O.

## CAPITULO II. ESTRUCTURA GENERAL DEL SISTEMA ELECTRONICO

### A. LA INTERFAZ PROPUESTA

#### A.1 INTRODUCCION

En este capítulo se explicará detalladamente el diseño de la interfaz paralela o tarjeta de expansión incluyendo además, las experiencias que determinaron la elección de este diseño para lograr que la lectora óptica se comunicara con cualquier bus de una PC-XT compatible con IBM.

#### A.2 ESTRUCTURA DEL SISTEMA

Para tener una clara idea de la estructura general del sistema, a continuación se presenta un diagrama a bloques de dicha configuración.

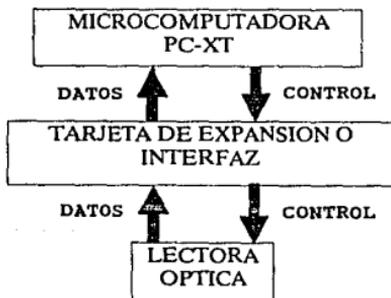


Figura 2.1 Estructura general del sistema

La estructura general del sistema consta de los siguientes elementos, tal como se muestra en la figura 2.1.

- \* Microcomputadora PC-XT compatible con IBM
- \* Tarjeta de expansión o interfaz
- \* Lectora óptica

Debido a que el sistema de control es una microcomputadora PC-XT y el cerebro de ésta es el microprocesador, tenemos lo siguiente.

Los microprocesadores 8088 de Intel trabajan en 2 diferentes modos de operación conocidos como modo mínimo y modo máximo. Si se quiere trabajar en colaboración con otros procesadores se debe elegir el modo máximo. El microprocesador 8088 contenido en la PC-XT trabaja en el modo de operación máximo, apoyándose de otros chips como el Controlador de Bus 8288, Coprocesador Matemático 8087, Coprocesador de Entrada/Salida 8089, etc.

Para mayores detalles relacionados a éste circuito, refierase al apéndice A.

### A.3 SEÑALES INVOLUCRADAS EN LA ESTRUCTURA GENERAL DEL SISTEMA

Lo primero que se contempló para determinar el hardware que se utilizó para el diseño de la tarjeta de interfaz está representado en la figura 2.1 como flechas que representan las señales que nos permiten la comunicación entre estos equipos.

De esta forma, se procedió al análisis de los diagramas electrónicos de la PC-XT y la lectora óptica para obtener la información de las señales de entrada y salida de los dos equipos.

Las señales involucradas en el ciclo de entrada/salida de la PC-XT se mostraron en el inciso B del capítulo anterior.

El resultado de la investigación para la lectora óptica nos permitió conocer la facilidad que tiene de recibir y enviar 16 bits de información, ambos en forma paralela.

La información que envía la lectora óptica al sistema de control a través de estos 16 bits nos permite conocer el estado de operación o error de la lectora óptica.

La información que recibe la lectora óptica del sistema de control es una serie de comandos de operación y funcionamiento.

#### A.4 COMANDOS Y ESTADOS DE OPERACION DE LA LECTORA OPTICA

A continuación se presentan los comandos y estados de operación de la lectora óptica:

COMANDO	NUMERO DE BIT															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JALAR HOJA	F	F	F	T	X	X	X	X	X	F	F	F	F	F	T	
MOTOR APAGADO	F	F	F	T	X	X	X	X	X	F	F	F	F	T	F	
DERIVACION 2	F	F	F	T	X	X	X	X	X	F	F	F	T	F	F	
DERIVACION 1	F	F	F	T	X	X	X	X	X	F	F	T	F	F	F	
LECTOR NO LISTO	F	F	F	T	X	X	X	X	X	F	T	F	F	F	F	
PET EDO DATOS DISPONIBLES	F	F	T	F	X	X	X	X	X	X	X	X	X	X	X	
PET PALABRA DE EDO # 1	F	F	T	T	X	X	X	X	X	X	X	X	X	X	X	
PET PALABRA DE EDO # 2	F	T	F	F	X	X	X	X	X	X	X	X	X	X	X	
PET PALABRA DE EDO # 3	F	T	F	T	X	X	X	X	X	X	X	X	X	X	X	
PET ENTRADA DE DATOS	F	T	T	F	X	X	X	X	X	X	X	X	X	X	X	
SELECCION DE LA PROM	F	T	T	T	X	X	X	X	X	X	C	X	A	A	A	
SELECCION BUFFER IMPRES	F	F	F	T	X	X	X	X	X	T	F	F	F	F	F	

DONDE:

	PC-XT	LECTOR OPTICO
T = VERDADERO	0 V	+ 5 V
F = FALSO	+ 5 V	0 V
X = CONDICION DE NO IMPORTA		
PET = PETICION		
EDO = ESTADO		
IMPRES = IMPRESION		

**TABLA COMPLEMENTARIA DE COMANDOS**

BIT #	1	0		C
<b>BITS DE SELECCION DE LA PROM</b>	<b>A</b>	<b>A</b>	<b>SIN RETARDO</b>	<b>F</b>
PROM # 1 UBICACION 15A	F	F	4 LINEAS RETARDO	T
PROM # 2 UBICACION 15B	F	T		
PROM # 3 UBICACION 15C	T	F		
PROM # 4 UBICACION 15D	T	T		

**NOTA:** SE TIENEN 4 PROMS PARA SER SELECCIONADAS Y LA FUNCION DE ESTAS ES LA DE INFORMAR A LA LECTORA OPTICA EL NUMERO DE PUNTOS DE EXPLORACION O LECTURA ENTRE RENGLONES QUE SE LLEVARAN A CABO.

LAS LINEAS DE RETARDO SE REFIEREN A LOS ESPACIOS ENTRE RENGLONES CONOCIDOS COMO MARCAS O MARGENES DE REGISTRO EN LAS CUALES LA LECTORA NO LEE INFORMACION.

**ESTADOS DE OPERACION**

BIT	PALABRA DE EDO #1	PALABRA DE EDO #2	PALABRA DE EDO #3	
			S U H	EDO DE ERROR
15	ERROR DE LECTURA	ERROR DE HOJA	0	1
14	I H VACIO	-	1	0
13	DEPOSITO A LLENO	-	0	0
12	DEPOSITO B LLENO	TORCIMIENTO R-TE	0	0
11	FALLA AL JALAR	TORCIMIENTO R-LE	0	0
10	MOTOR APAGADO	TORCIMIENTO F-TE	0	0
9	HOJAS MULTIPLES	TORCIMIENTO F-LE	0	0
8	DEPOSITO A TARDIO	LONGITUD CKF	0	0
7	DEPOSITO B TARDIO	LONGITUD CKR	0	0
6	VERIFIQUE LECTURA	IMPRESION TARDIA	0	0
5	ATASCAMIENTO	CORRIMIENTO LR 5	0	0

4	AUTO-PRUEBA	CORRIMIENTO LR 4	0	0
3	SUH - FRENTE	CORRIMIENTO LR 3	SUHF	SUHF
2	SUH - REVERSO	CORRIMIENTO LR 2	SUHR	SUHR
1	HOJA EN TRANSITO	CORRIMIENTO LR 1	0	0
0	CUBIERTA ABIERTA	CORRIMIENTO LR 0	0	0

**DONDE:**

**PALABRA DE EDO #1:** Errores de hardware y condiciones de lector no-listo.

**PALABRA DE EDO #2:** Dato de hoja desviada.

**PALABRA DE EDO #3:** Condición Sheet Under Head. (El dato de lectura esta en existencia).

**I H:** Entrada del alimentador de hojas de lectura.

**F:** Frente de la hoja de lectura.

**R:** Reverso de la hoja de lectura.

**R-TE:** Reverso lado derecho.

**R-LE:** Reverso lado izquierdo.

**F-TE:** Frente lado derecho.

**F-LE:** Frente lado izquierdo.

**CKF:** Revisión del frente de la hoja.

**CKR:** Revisión del reverso de la hoja.

**SUH:** Hoja bajo la parte superior.

**LR:** Izquierda a derecha.

**B. DISEÑO DE LA TARJETA DE EXPANSION O INTERFAZ.**

**B.1 INTRODUCCION**

Cuando se desea diseñar una tarjeta de expansión o interfaz para una computadora personal se requiere, en primera instancia,

conocer las señales que nos proporciona el bus de expansión de la computadora, así como el mapa de líneas de entrada/salida; esto con el fin de localizar aquellas líneas de direccionamiento que se encuentran disponibles para trabajar con tarjetas de expansión.

Una vez conocidas las señales de entrada/salida de la PC-XT y de la lectora óptica, el siguiente paso fué la elección de los componentes de la tarjeta de interfaz.

## B.2 ELECCION DE LOS COMPONENTES

Para lograr la comunicación entre la PC-XT y la lectora óptica hay que interconectar los buses de datos, direcciones y de control, como primer paso.

Se requiere además de dispositivos latches que nos permiten tomar estos datos para ser reconocidos tanto por la lectora como por la PC-XT.

Las alternativas fueron un chip 74LS373 y un PPI8255A.

La decisión favoreció al PPI8255A, fundamentalmente porque éste es un dispositivo "inteligente" ya que es un coprocesador programable, de tal forma que al microprocesador de la PC-XT se le libera de las tareas de control del puerto de entrada/salida que nos permite tener un programa de control más sencillo.

Debido a que se manejan 16 bit de datos de entrada y otros tantos de salida, se utilizaron 2 PPI's. Además se utilizaron un tranceptor, un decodificador y circuitería de decodificación.

El tranceptor nos permite la comunicación bidireccional de datos entre el bus de expansión de la PC-XT y los PPI's.

Para la decodificación de direcciones se utilizó lógica combinacional conectada a las entradas del decodificador y las salidas de éste se conectaron directamente al pin de selección "CS" de cada uno de los PPI's, con el objeto de seleccionar el PPI adecuado, es decir, el de entrada o de salida.

Los dos PPI's se programaron en el modo cero, ya que este modo permitió habilitar los puertos A y B como un solo bus de 16 bits para cada chip. Esto implica que un circuito se habilitó como puerto de entrada de 16 bits y el otro circuito como puerto de salida, también de 16 bits.

Debido a que este circuito nos permite la comunicación de la PC-XT con la lectora óptica, y en general, nos permite la comunicación entre el bus de expansión de la PC-XT y cualquier equipo periférico, a continuación se describirá ampliamente.

### B.3 CIRCUITO PPI 8255A

El circuito PPI (Interfaz Periférica Programable) 8255A de Intel es un dispositivo programable de entrada/salida de propósito general diseñado para ser usado con microprocesadores Intel.

Tiene 24 terminales de entrada/salida las cuales se pueden programar individualmente en 2 grupos de 12 y se puede utilizar en 3 diferentes modos de operación.

La figura 2.2 nos muestra el patigrama y la estructura interna de este dispositivo.

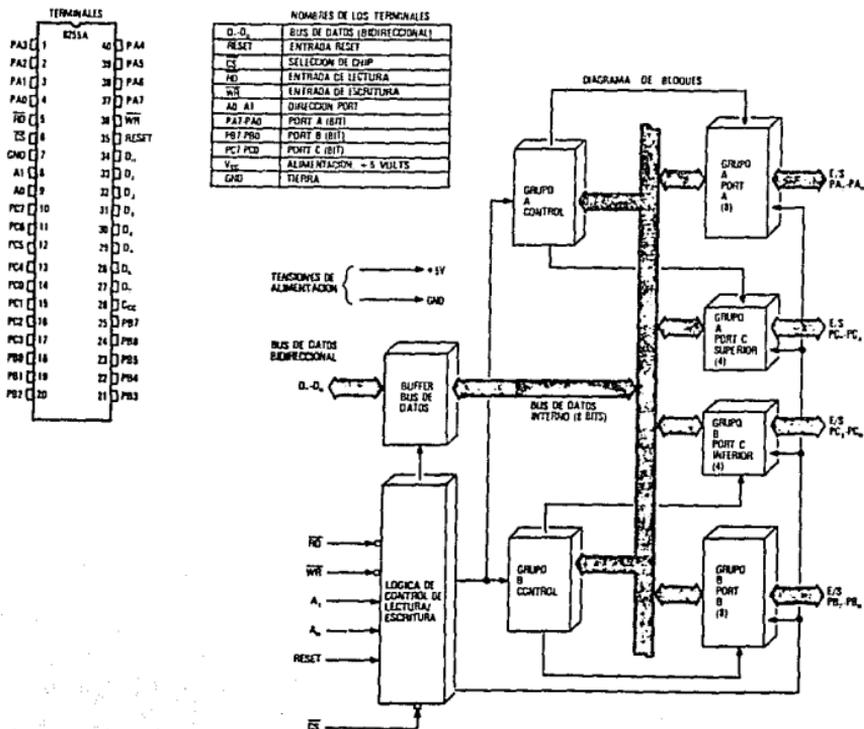


Figura 2.2 Patigrama y estructura interna del PPI 8255A

### Buffer del Bus de Datos

Este buffer bidireccional de 8 bits y con la característica de tener tercer estado en sus terminales se utiliza para conectar el PPI al bus de datos del sistema. Los datos son transmitidos o recibidos por el buffer a través de las instrucciones del CPU, siendo éstas de entrada o de salida. Las palabras de control y el estado de la información también se transmiten a través del buffer.

### R/W y Control Lógico

La función de este bloque es manejar todas las transferencias internas y externas relacionadas con los Datos, Control o Palabras de Estado. Acepta entradas provenientes de los buses de Control y Direcciones del CPU y emite comandos a los grupos de control.

### $\overline{CS}$

Selección del chip. Un bajo en la entrada de esta terminal, habilita la comunicación entre el 8255A y el CPU.

### $\overline{RD}$

Lectura. Un bajo en la entrada de esta terminal, habilita al 8255A para enviar el dato o estado de la información al CPU a través del bus de datos. Esto permite al CPU leer del 8255A.

### $\overline{WR}$

Escritura. Un bajo en la entrada de esta terminal, habilita al CPU para escribir datos o palabras de control al 8255A.

### A0 y A1

Puertos de selección 0 y 1. Estas señales de entrada, en conjunto con las señales RD y WR, controlan la selección de uno de los tres puertos o los registros de la palabra de control. Normalmente se conectan a los bits menos significativos del bus de direcciones (A0 y A1).

### Reset

Reinicialización. Un alto en la entrada de esta terminal limpia el registro de control y todos los puertos (A, B, C) son colocados en el modo de entrada.

## Control de Grupo A y B

La configuración funcional de cada puerto es programado por el software del sistema. Es decir, las salidas del CPU van a la palabra de control del 8255A. La palabra de control contiene información tal como "modo", "bit de habilitación", "bit de rehabilitación", etc., que inicia la configuración funcional del 8255A.

Cada uno de los bloques de control (Grupo A y Grupo B) acepta comandos del control lógico R/W, recibe palabras de control del bus de datos interno y envía los comandos apropiados para los puertos asociados.

Control de Grupo A - Puerto A y Puerto C [superior (C7 - C4)]

Control de Grupo B - Puerto B y Puerto C [inferior (C3 - C0)]

El registro de la palabra de control puede solamente ser escrito.

No se permite la operación de lectura de la palabra de control.

## Puertos A, B, C

El 8255A contiene 3 puertos (A, B, C) de 8 bits cada uno. Todos pueden ser configurados en una amplia variedad de características funcionales a través del software del sistema, pero cada uno tiene sus características especiales.

**Puerto A.** Una salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits latch.

**Puerto B.** Una entrada/salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits buffer.

**Puerto C.** Una salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits buffer unicamente. Este puerto puede ser dividido en dos puertos de 4 bits bajo el modo de control. Cada puerto de 4 bits contiene un latch de 4 bits y este puede ser usado para el control de las señales de estado de entrada en conjunto con los puertos A y B.

## Selección del modo

Hay tres modos básicos de operación que pueden ser seleccionados por el software del sistema:

Modo 0. Modo básico de entrada/salida.

Modo 1. Modo de "Strobe" de entrada/salida.

## Modo 2. Modo de bus bidireccional.

El modo 0 para la programación del PPI 8255A es el modo básico de entrada/salida.

La palabra de control de programación de modo consta de 8 bits que van desde D0 hasta D7, donde D0 es el bit menos significativo (LSB) y D7 es el bit más significativo (MSB).

Como puede observarse en la figura 2.3 se muestra la estructura que conforma la palabra de control de programación del PPI 8255A.

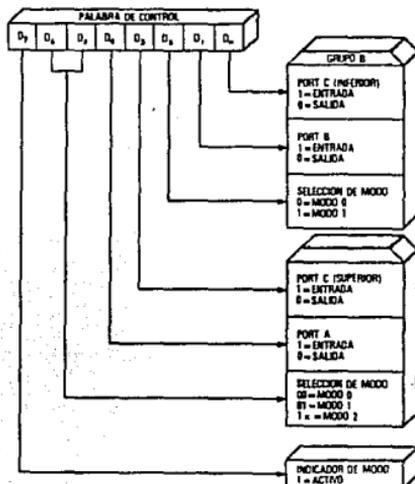


Figura 2.3 Palabra de control para programación del PPI 8255A

A continuación se especifica cada uno de los bits que conforman la palabra de programación.

- |     |   |                           |
|-----|---|---------------------------|
| D0: | Pertenece al Grupo B.<br>Puerto C (inferior): | 1 = entrada<br>0 = salida |
| D1: | Pertenece al Grupo B.<br>Puerto B:            | 1 = entrada<br>0 = salida |

D2: Pertenece al Grupo B.  
 Selección del modo:            0 = modo 0  
    1 = modo 1

D3: Pertenece al Grupo A.  
 Puerto C (superior):            1 = entrada  
    0 = salida

D4: Pertenece al Grupo A.  
 Puerto A:                            1 = entrada  
    0 = salida

D6, D5: Pertenece Grupo A.  
 Selección del modo:            00 = modo 0  
    01 = modo 1  
    1X = modo 2

D7: Modo de colocación de la bandera:            1 = activo

Las palabras de control utilizadas para programar los 2 PPI's 8255A se muestran a continuación:

**PALABRA DE CONTROL DEL PUERTO DE ENTRADA:**

D7	D6	D5	D4	D3	D2	D1	D0		
1	0	0	1	1	0	1	1	=	9 B H

**PALABRA DE CONTROL DEL PUERTO DE SALIDA:**

D7	D6	D5	D4	D3	D2	D1	D0		
1	0	0	0	0	0	0	0	=	8 0 H

**CARACTERISTICA DE INICIALIZACION Y REINICIALIZACION "SINGLE BIT"**

Uno de los 8 bits del puerto C puede ser inicializado o reinicializado usando la instrucción de salida sencilla. Esta característica reduce los requerimientos de software en aplicaciones basadas en control. Cuando el puerto C es usado como estado/control del puerto A o B, estos bits pueden ser inicializados o reinicializados usando unicamente la operación del bit de inicialización/reinicialización como si fueran datos de puerto de salida.

**B.4 DIRECCIONES**

Las líneas de direcciones de la A0 a la A9 nos permiten

seleccionar la tarjeta de expansión o interfaz montada sobre el bus de la PC-XT, comprendiendo un KB de espacio reservado para localidades estandar de entrada/salida.

## **C. IMPLEMENTACION DE LA INTERFAZ PROPUESTA ENTRE LA LECTORA OPTICA Y LA PC-XT**

### **C.1 INTRODUCCION**

En este punto se presentara el diseño electrónico final de la interfaz entre la lectora óptica y la PC-XT.

### **C.2 JUSTIFICACION DE LA INTERFAZ**

Como puede observarse en el diagrama de la figura 2.4, se tiene un diseño relativamente sencillo para lograr la interfaz de comunicación entre la lectora óptica y la PC-XT.

El diseño de la interfaz no presento mayor problema, pero lo importante y complicado del asunto fué el hecho de acoplar las señales provenientes de la lectora óptica a la PC-XT y viceversa.

Se invirtieron muchas horas de estudio, análisis y pruebas, debido a que la PC-XT trabaja con una velocidad de reloj de 4.77 MHz, mientras que la lectora óptica trabaja a una velocidad de reloj de 10 MHz.

Cada Renglón de lectura se compone de 10 líneas y cada línea contiene 1024 pixels, además se leen 16 palabras de datos de 16 bits por renglón.

De las pruebas realizadas se determinó que por cada lado de la hoja se tiene un total de 61 renglones, que multiplicados por los 16 datos por renglón nos da un total de 976 datos por lado de la hoja ó 488 bytes por lado de la hoja. Por tanto se tiene un total de 1952 palabras de datos de 16 bits por hoja ó 976 bytes por hoja.

El proceso de lectura baja considerablemente la velocidad de la lectora óptica y teóricamente se leen 85 hojas por minuto, aproximadamente (1.4 hojas por segundo).

Otro punto importante de considerar fué el encontrar la decodificación de la lectora óptica al transmitir los datos a la PC-XT con la finalidad de desarrollar el programa de aplicación.

En este punto se pudo determinar que cada albeolo u óvalo de información formado por 9 pixels adyacentes se representa con 4 bits y se le conoce como canal. Cada renglón contiene 64 canales.

En la práctica se utilizan dos albeolos unidos para formar una posición de respuesta que es la que la lectora reporta como marca de información a la PC-XT que será evaluada por el programa de aplicación.

Definir el rango de localidades de memoria para el almacenaje de la información, así como definir los tiempos para dicha labor origino también algunos problemas. La información del lado frontal de la hoja se almacena en un espacio de localidades de memoria de 07A0H, mientras que la información del lado posterior de la hoja se almacena en un espacio de localidades de memoria de 7AFH.

La lectora óptica realiza la lectura de izquierda a derecha y de arriba hacia abajo. Esta información la almacena en localidades de memoria sucesivas en el siguiente formato:

X X X X B E D 3 X X X X

Donde:

B, E, D, 3, X son valores de nivel de intensidad.

E: Primer valor leído.

B: Segundo valor leído.

3: Tercer valor leído.

D: Cuarto valor leído.

El nivel de intensidad claro-oscuro se muestra a continuación.

VALOR HEXA	NIVEL
0 - 6	H
7 - 9	M
B - C	L
D, E, F	NINGUNO

Donde:

H: Marca oscura.

M: Marca intermedia.

L: Marca Clara.

Ninguno: Ausencia de marca.

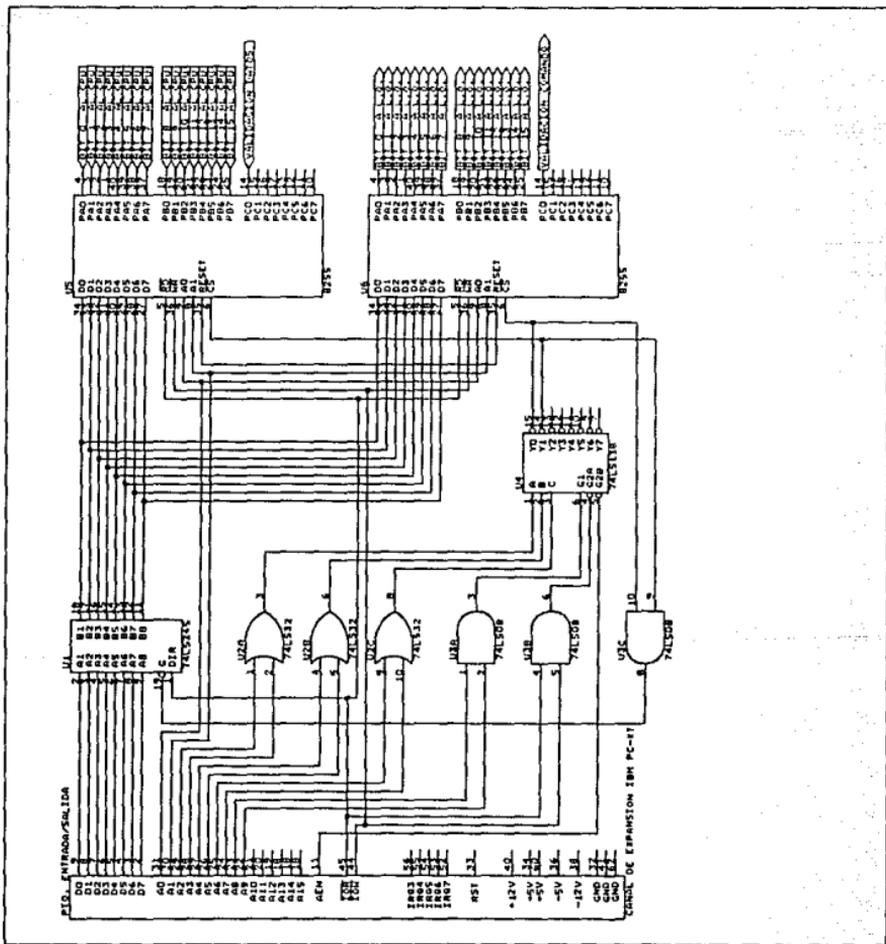


figura 2.4 Diagrama electrónico de la tarjeta de interfaz

#### D. DISEÑO DEL CIRCUITO IMPRESO

El proyecto inicialmente se implementó en Tabletas Protoboard.

Una vez concluida la parte de pruebas, el siguiente paso de la realización física de la interfaz electrónica, consistió en transferir todo el diseño de esta a circuito impreso. Esto se llevo a cabo con el paquete de computadora Smartwork.

Tomando la filosofía de las tarjetas de las PC en general, las cuales se interconectan al bus de expansión en forma modular, se diseñó la tarjeta con tales características.

Para la elaboración del circuito impreso se considero la distribución de los circuitos en base a sus características, función desempeñada e interconexión con los demás elementos, ya que de esto dependen el tamaño, funcionalidad y demás características de la tarjeta.

## CAPITULO III. PROGRAMA DE CONTROL DE LA INTERFAZ

### A. DISEÑO DEL DIAGRAMA DE FLUJO

#### A.1 INTRODUCCION

En el presente capítulo se muestra el programa de control para el manejo de la lectora óptica a través de la PC-XT compatible con IBM.

El primer paso para elaborar el programa de control fué el de diseñar un diagrama de flujo que nos permitiera definir la secuencia de instrucciones que ejecuta la lectora óptica con objeto de optimizar el número de instrucciones del programa de control.

Una vez definido el diagrama de flujo, se procedió a la elaboración del programa principal, así como las subrutinas de uso especial que soportan a dicho programa.

El lenguaje utilizado para el diseño del programa fué lenguaje ensamblador.

#### A.2 CONSIDERACIONES FINALES

Para definir el diagrama de flujo se analizó el procedimiento lógico que lleva a cabo la lectora óptica para leer y transmitir la información al sistema de control.

Los puntos relevantes de este procedimiento se describen a continuación.

1. Para inicializar el **buffer** de la lectora óptica hay que preguntarle por su palabra de estado # 1.
2. Cuando termina de leer el lado frontal de la hoja se le debe preguntar nuevamente por la palabra de estado # 1 para inicializar los contadores y los **buffers** de la lectora.
3. Si no se cumple el punto anterior, la información que envía la lectora óptica será diferente a la información que recibe el sistema de control.
4. Cuando se da un comando a la lectora óptica en el cual no se espera respuesta de información sino respuesta de ejecución, es decir, desactivar o activar la bandera de control de los puertos PPI 8255A para reportar la presencia de información en los **buffers** de la lectora, se requiere darle un retardo por **software**

al programa para asegurar que el dato que envia la lectora es el dato que captura el sistema de control.

### A.3 DIAGRAMA DE FLUJO

A continuación se presenta el diagrama de flujo general que nos permitió estructurar el programa de control de la lectora óptica.

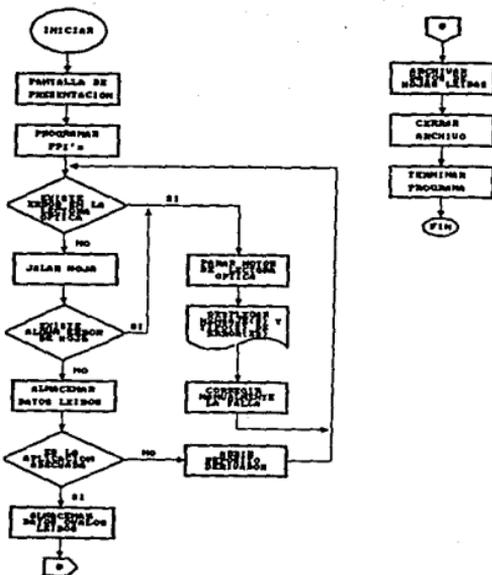


FIGURA 2.1 DIAGRAMA DE FLUJO GENERAL

El diagrama de flujo particular nos muestra con lujo de detalle la secuencia de pasos de control de la lectora óptica.

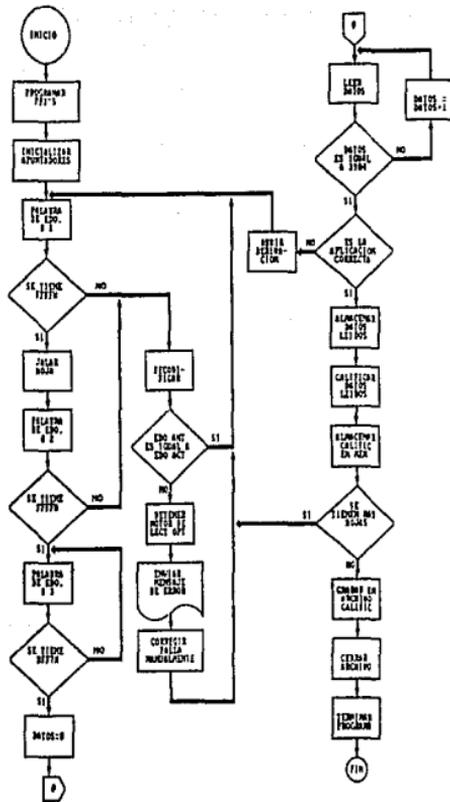


FIGURA 3.2 DIAGRAMA DE FLUJO PARTICULAR

NOTA:

FFFFH: Palabra de condición de funcionamiento correcto al solicitar la palabras de estado número 1 y 2.

BFF7H: Palabra de condición de lectura del lado frontal de la hoja.

B. LISTADO DEL PROGRAMA A EJECUTAR

;PROGRAMA DE CONTROL PARA LECTORA OPTICA

```

dat_sal equ 0300h ;Dir. pto. de sal. de datos
val_sal equ 0302h ;Dir. pto. validar datos sal.
ctr_sal equ 0303h ;Dir. pto. de control sal.
dat_ent equ 0304h ;Dir. pto. de ent. de datos
val_ent equ 0306h ;Dir. pto. validar datos ent.
ctr_ent equ 0307h ;Dir. pto. de control ent.

```

```

;*****
datos segment ;Definir segmento de datos

```

```

include datos.tes ;Datos de programa

```

```

datos ends
;*****

```

```

lect_opnam segment

```

```

;-----
princ proc far ;Definir segmento de codigo

```

```

assume cs:lect_opnam,ds:datos

```

```

inicio: ;Dir. inicio de ejecución

```

```

;Actualizar stack para regresar a DOS

```

```

push ds ;Escribir posición de reg DS
sub ax,ax ;Inicializar cero en AX
push ax ;Escribir desplaz cero en el Stack

```

```

;Igualar reg DS al actual Segmento de datos

```

```

mov ax,datos ;Cargar datos en AX
mov ds,ax ;y cargar en reg DS datos

```

```

mov hojas,00h ;Inicializar # de hojas calificadas
mov di,0000 ;Inicializar desplaz de mem de datos de evaluaciones
call presenta ;Llamar presentación de programa

```

```

cmp prog_no,0ffh ;Es aplicación correcta ?
jz nada ;No, terminar programa

```

```

call archivo ;Si, Pedir archivo donde almacenar info
;Y asignar parametros a este
call crlf ;Retornar y avanzar línea

```

```

call ppi ;Programar PPI de ent y sal

```

```

call prom ;Seleccionar prom "15A"

```

```

otra_hoja:

```

```

call esdo_1 ;Preguntar edo 1 a L O y enviar
;letrero correspondiente
call pick ;Extraer una hoja

```

```

call esdo_2 ;Preguntar edo 2 a L 0 y enviar
;letrero correspondiente
call esdo_3 ;Preguntar edo 3 a L 0 y enviar
;letrero correspondiente
call pet_dat ;Pedir datos de hoja leida

call dat_mem ;Almacenar espejo de hoja leida
;en memoria de PC
call aplic_? ;Comprobar aplicación seleccionada

cmp apli_no,01;La hoja pertenece a aplicación correcta ?
jnz guarda ;Si, continuar programa

der_abr:
call retraso ;No, realizar retardo

call abr_der ;Abrir derivación

call retraso ;Realizar retardo

jmp otra_hoja ;Ir a pedir edo 1

guarda:
push di ;Escribir en stack desplaz de memoria de
;datos de evaluaciones
call fol_cal ;Almacenar en memoria de PC contenido de
;los ovalos con info. de folio y califs.
call recal ;Almacenar contenido de ovalos con info de
;corrección
pop di ;Regresar a di desplaz mem de evaluaciones

call aplica ;Almacenar en mem contenido de ovalos
;seleccionados por el usuario
cr_lf ;enviar un retorno y avance de línea
;cada hoja calificada
call edo_1 ;Pedir edo 1 a L 0 en ax
and ax,4000h ;Enmascarar los 4 bits más significativos
cmp ax,4000h ;Hay hojas en alimentador ?
jz otra_hoja ;Si, Jalar otra hoja

call write ;No, Escribir Información

call terminar ;Finalizar programa

nada:
ret ;Terminar prog

princ endp ;Fin parte principal del programa
;*****
;Proc. presentar pantalla de inicio
;presenta proc near

include presenta.tes ;Incluir pantalla de presentación

ret ;Regresar

presenta endp ;Fin de proc presenta

```

```

;-----
include arch.tes ;Rutinas manejo de archivo
;-----
include coman.tes ; Llamar rutinas de comandos para lect. opt.
;-----
include error.tes ;Manejar errores de lect. Optica
;-----
include rut's.tes ;Rutinas adicionales
;-----
include aplic.tes ;Manejo de informacion leida por lect. opt.
;-----
;Terminar prog
Terminar proc near
    include terminar.tes
    ret ;Regresar
terminar endp ;Fin de proc. terminar
;-----
lect_opnam ends ;Fin de segmento de codigo
;*****
end inicio ;Fin de ensamblado

```

;DATOS DEL PROGRAMA DE CONTROL DE LECTORA OPTICA

-----  
espejo dw 1959 dup(' '), '\$' ; Locs para almacenar espejo de la hoja

-----  
;Datos para presentar pantalla

unam db 'UNIVERSIDAD NACIONAL AUTONOMA DE MEXICOS'

dgscaa db ' DIRECCION DE SERVICIOS DE COMPUTO PARA LA ADMINISTRACION ACADEMICAS'

let db ' RESPALDO DE INFORMACION DE LECTURA OPTICAS'

let\_0 db 'Aplicación: ACTAS DE EXAMENS'

let\_1 db '1) Admisión a Bachillerato 5) Ordinario\$'

let\_2 db '2) Admisión a Licenciatura 6) Extraordinario\$'

let\_3 db '3) Reinscripción a Bachillerato 7) Colocación CELE-UNAMS'

let\_4 db '4) Reinscripción a Licenciatura 8) Evaluación Medica\$'

selec db 'Seleccione la opción que desea calificar: \$'

no\_corre db ' \*\*\* ESTA APLICACION NO ESTA DISPONIBLE \*\*\*\$'

selec\_no db ' \*\*\* SU SELECCION NO ES CORRECTA \*\*\*\$'

prog\_no db ? ;Variable que indica si la aplicación seleccionada  
;esta disponible o no

-----  
;Datos para pedir archivo donde se va a almacenar la  
;información y definición de parámetros de este archivo

arch db 0dh,0ah, 'ARCHIVO DONDE DESEA ALMACENAR SU INFORMACION: \$'

nombuff db 50 ;Bytes máximos para trayectoria

No\_arch dw ? ;Número de archivo creado

nom\_arch db 'SU INFORMACION FUE ALMACENADA EN EL ARCHIVO: \$'

hoj\_cal db 'NUMERO DE HOJAS CALIFICADAS: \$'

hojas dw ? ;Variable que almacena el # de hojas calificadas

-----  
info db 249 dup(' '), '\$' ;Locs para almacenar los ovals  
;que contienen información

-----  
;Posición de las marcas correspondientes a la aplicación

marca dw 74,146,362,650,794,866,1226,1370,3674,3746,3962,4250,4322

-----  
mas\_hoja db ? ;Variable para almacenar el parámetro que indica  
;si la hoja corresponde a la aplicación seleccionada

;Posición de Ovalo correspondiente al folio

```

No_ova dw 832, 904, 976, 1048, 1120, 1192, 1264, 1336, 1408, 1480
dw 834, 906, 978, 1050, 1122, 1194, 1266, 1338, 1410, 1482
dw 838, 910, 982, 1054, 1126, 1198, 1270, 1342, 1414, 1486
dw 842, 914, 986, 1058, 1130, 1202, 1274, 1346, 1418, 1490
dw 846, 918, 990, 1062, 1134, 1206, 1278, 1350, 1422, 1494
dw 850, 922, 994, 1066, 1138, 1210, 1282, 1354, 1426, 1498
dw 854, 926, 998, 1070, 1142, 1214, 1286, 1358, 1430, 1502
    
```

;Posición de oвалos correspondientes a las calificaciones

```

dw 1704, 1708, 1710, 1714, 1716, 1776, 1780, 1782, 1786, 1788
dw 1848, 1852, 1854, 1858, 1860, 1920, 1924, 1926, 1930, 1932
dw 1992, 1996, 1998, 2002, 2004, 2064, 2068, 2070, 2074, 2076
dw 2136, 2140, 2142, 2146, 2148, 2208, 2212, 2214, 2218, 2220
dw 2280, 2284, 2286, 2290, 2292, 2352, 2356, 2358, 2362, 2364
dw 2424, 2428, 2430, 2434, 2436, 2496, 2500, 2502, 2506, 2508
dw 2568, 2572, 2574, 2578, 2580, 2640, 2644, 2646, 2650, 2652
dw 2712, 2716, 2718, 2722, 2724, 2784, 2788, 2790, 2794, 2796
dw 2856, 2860, 2862, 2866, 2868, 2928, 2932, 2934, 2938, 2940
dw 3000, 3004, 3006, 3010, 3012, 3072, 3076, 3078, 3082, 3084
dw 3144, 3148, 3150, 3154, 3156, 3216, 3220, 3222, 3226, 3228
dw 3288, 3292, 3294, 3298, 3300, 3360, 3364, 3366, 3370, 3372
dw 3432, 3436, 3438, 3442, 3444
    
```

;Posición de ovalos correspondientes a las correcciones

```

dw 3607, 3609, 3611, 3679, 3681, 3683, 3685, 3687, 3689, 3691
dw 3693, 3695, 3697, 3720, 3724, 3726, 3730, 3732, 3751, 3753
dw 3755, 3823, 3825, 3827, 3829, 3831, 3833, 3835, 3837, 3839
dw 3841, 3864, 3868, 3870, 3874, 3876, 3895, 3897, 3899, 3967
dw 3969, 3971, 3973, 3975, 3977, 3979, 3981, 3983, 3985, 4008
dw 4012, 4014, 4018, 4020
    
```

```

-----
evalua db 21000 dup (' ', ' ');Localidades para almacenar la
;calificación de 300 hojas máximo
    
```

```

apli_no db ? ;Variable para comprobación de existencia de folio
    
```

```

status1 dw ? ;Variable asignada a palabra de estado 1 de L O
    
```

```

status2 dw ? ;Variable asignada a palabra de estado 2 de L O
    
```

```

status3 dw ? ;Variable asignada a palabra de estado 3 de L O
    
```

```

-----
;Mensajes de error enviados por la L O debidos a estado 1
    
```

```

UNO_15 db "!!!!!!! LECTOR NO LISTO !!!!!!!"
    
```

```

UNO_14 db "!!!!!!! ALIMENTADOR DE HOJAS VACIO !!!!!!!"
    
```

```

UNO_13 db "!!!!!!! DERIVACION A LLENO !!!!!!!"
    
```

```

UNO_12 db "!!!!!!! DERIVACION B LLENO !!!!!!!"
    
```

UNO\_11 db "||||||| FALLA AL JALAR HOJA |||||"  
 UNO\_10 db "||||||| MOTOR APAGADO |||||"  
 UNO\_9 db "||||||| HOJAS MULTIPLES |||||"  
 UNO\_8 db "||||||| ABERTURA DE DERIVACION A TARDIA |||||"  
 UNO\_7 db "||||||| ABERTURA DE DERIVACION B TARDIA |||||"  
 UNO\_6 db "||||||| VERIFIQUE LECTURA |||||"  
 UNO\_5 db "||||||| ATASCAMIENTO |||||"  
 UNO\_4 db "||||||| AUTOPRUEBA |||||"  
 UNO\_3 db "||||||| SUH - FRENTE |||||"  
 UNO\_2 db "||||||| SUH - REVERSO |||||"  
 UNO\_1 db "||||||| HOJA EN TRANSITO |||||"  
 UNO\_0 db "||||||| CUBIERTA ABIERTA |||||"

;.....  
 ;Mensajes de error enviados por la Lectora Op. debidos a estado 1

DOS\_15 db "||||||| ERROR DE HOJA |||||"  
 DOS\_12 db "||||||| TORCIMIENTO EN EL REVERSO DEL LADO DERECHO |||||"  
 DOS\_11 db "||||||| TORCIMIENTO EN EL REVERSO DEL LADO IZQUIERDO |||||"  
 DOS\_10 db "||||||| TORCIMIENTO EN EL FRENTE DEL LADO DERECHO |||||"  
 DOS\_9 db "||||||| TORCIMIENTO EN EL FRENTE DEL LADO IZQUIERDO |||||"  
 DOS\_8 db "||||||| REVISION DE LONGITUD DEL FRENTE DE LA HOJA |||||"  
 DOS\_7 db "||||||| REVISION DE LONGITUD DEL FRENTE DE LA HOJA |||||"  
 DOS\_6 db "||||||| IMPRESION TARDIA |||||"  
 DOS\_5 db "||||||| CORRIMIENTO 5 DE IZQUIERDA A DERECHA DE |||||"  
 DOS\_4 db "||||||| CORRIMIENTO 4 DE IZQUIERDA A DERECHA DE |||||"  
 DOS\_3 db "||||||| CORRIMIENTO 3 DE IZQUIERDA A DERECHA DE |||||"  
 DOS\_2 db "||||||| CORRIMIENTO 2 DE IZQUIERDA A DERECHA DE |||||"  
 DOS\_1 db "||||||| CORRIMIENTO 1 DE IZQUIERDA A DERECHA DE |||||"  
 DOS\_0 db "||||||| CORRIMIENTO 0 DE IZQUIERDA A DERECHA DE |||||"

;.....  
 ;Mensajes de error enviados por la L O debidos a estado 3

SUHF\_3 db "||||||| ERROR FRENTE SUPERIOR DE LA HOJA |||||"  
 SUHR\_3 db "||||||| ERROR REVERSO SUPERIOR DE LA HOJA |||||"

C. SUBROUTINAS DE USO ESPECIAL

```

mov prog_no,00
;-----
;Borrar pantalla

mov ah,06h ;Cargar función de avanzar pantalla en ah
mov al,00 ;Código para limpiar pantalla
mov cx,0000 ;Renglón y columna de inicio
mov dx,2479h ;Renglón y columna final
mov bh,07h ;Poner atributo normal
int 10h ;Llamar a ROM video
;-----
;Posicionar cursor en parte superior pantalla

mov ah,02 ;Función posicionar cursor
mov dh,00 ;Renglón de inicio
mov dl,00 ;Columna de inicio
mov bh,00 ;Página actual
int 10h ;Llamar a ROM video
;-----
;Enviar letrero unam

mov dx,offset unam ;Mensaje unam
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS
;-----
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
;-----
;Enviar letrero dgscas

mov dx,offset dgscas ;Mensaje dgscas
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
;-----
;Enviar letrero let

mov dx,offset let ;Mensaje let
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
;-----

```

;Enviar letrero de aplicaciones

```
mov dx,offset let_0 ;mensaje let_0
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea

mov dx,offset let_1 ;mensaje let_1
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea

mov dx,offset let_2 ;Mensaje let_2
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea

mov dx,offset let_3 ;Mensaje let_3
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea

mov dx,offset let_4 ;Mensaje let_4
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS

call crlf ;Retornar y avanzar línea
```

otra\_sel:

-----  
;Enviar letrero para seleccionar aplicación

```
mov dx,offset selec ;Mensaje selec
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS
```

-----  
;Pedir selección de aplicación

```
mov ah,01 ;Función entrada teclado
int 21h ;Llamar a DOS

push ax ;Escribir ax en stack

call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea

pop ax ;Leer ax de stack
```

-----

;Discriminar aplicación

```
mov ah,00 ;Ceros en ah
cmp ax,0030h ;Se seleccionó opción no válida ?
jl no_opc ;Si, enviar letrero de aviso
cmp ax,003Bh ;Se seleccionó opción no válida ?
jg no_opc ;Si, enviar letrero de aviso

cmp al,'1' ;Opción 1 ?
jz no_prog ;Si, ir a no_prog
cmp al,'2' ;Opción 2 ?
jz no_prog ;Si, ir a no_prog
cmp al,'3' ;Opción 3 ?
jz no_prog ;Si, ir a no_prog
cmp al,'4' ;Opción 4 ?
jz no_prog ;Si, ir a no_prog
cmp al,'5' ;Opción 5
jz si_prog ;Si, ir a si_prog
cmp al,'6' ;Opción 6 ?
jz si_prog ;Si, ir a si_prog
cmp al,'7' ;Opción 7 ?
jz no_prog ;Si, ir a no_prog
cmp al,'8' ;Opción 8 ?
jz no_prog ;Si, ir a no_prog
```

-----  
;Enviar letrero selección no adecuada

```
no_opc:
mov dx,offset selec_no ;Mensaje avis_no
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS
```

-----  
;Posicionar cursor en petición de aplicación

```
mov ah,02 ;Función posicionar cursor
mov dh,18d ;Renglón de inicio
mov dl,00d ;Columna de inicio
mov bh,00 ;Página actual
int 10h ;Llamar a ROM video
```

```
jmp otra_sel ;Preguntar por otra aplicación
```

no\_prog:

-----  
;Enviar letrero de no aplicación

```
mov dx,offset no_corre ;Mensaje no_corre
mov ah,09h ;Función imprimir mensaje
int 21h ;Llamar a DOS
```

```
call crlf ;Retornar y avanzar línea
call crlf ;Retornar y avanzar línea
```

```
mov prog_no,0ffh;Avisar no aplicación
jmp sin_pro
```

;Borrar pantalla a partir de renglón 13h hacia abajo

si\_prog:

```
mov ah,06h ;Función avanzar pantalla
mov al,00 ;Código limpiar pantalla
mov cx,1300h ;Renglón y columna de inicio
mov dx,2479h ;Renglón y columna final
mov bh,07h ;Poner atributo normal
int 10h ;Llamar a ROM video
```

sín\_pro:

-----  
;Proc para crear archivo donde almacenar info calificada  
archivo proc near

```
mov dx,offset arch ;Mensaje pedir archivo
mov ah,9h ;Función imprimir mensaje
int 21h ;Llamar a DOS

mov dx,offset nonbuff ;Dir del buffer
mov ah,0ah ;Función ent de teclado al buffer
int 21h ;Llamar a DOS
```

;Insertar cero en buffer seguido del nombre del archivo

```
mov bl,nonbuff+1 ;Obtener # de bytes leídos
mov bh,0 ;Poner en bx
mov [nonbuff+bx+2],0 ;Cero después del nombre del
;archivo (ultimo byte)
```

;Crear archivo

```
mov dx,offset nonbuff+2 ;Dir del nombre del arch
mov cx,0 ;Atributo normal
mov ah,3ch ;Función crear archivo
int 21h ;Llamar a DOS
mov No_arch,ax ;Almacenar # archivo

ret ;Regresar
```

archivo endp ;Fin proc archivo

-----  
;Escribir en un archivo info calificada y cerrar el archivo  
write proc near

```
mov bx,No_arch;Enviar # de archivo a bx
mov dx,offset evalua ;Dir buffer
mov cx,d1 ;# de bytes a escribir
mov ah,40h ;Función de escritura a archivo
int 21h ;Llamar a DOS
```

;Cerrar archivo

```
mov bx,No_arch;Obtener # de archivo en bx
mov ah,3eh ;Función cierre de archivo
int 21h ;Llamar a DOS

ret ;Regresar a prog princ
```

write endp ;Fin proc write

-----

```

-----
;Proc. seleccionar PROM "15A"
prom   proc near

        call cambio   ;Bandera validar datos
                        ;a 1 (desactivada)
        mov  ax,8ff3h ;Seleccionar prom "15A"

        call env_dat  ;Enviar selección y
                        ;validar envío
        call cambio   ;Bandera validación de
                        ;datos a 1 (desactivada)

        call retardo  ;Hacer retardo

        ret           ;Regresar a prog princ

prom   endp          ;Fin proc prom
-----
;Proc leer y almacenar datos leídos de L 0
dat_mem proc near

mem_dat:
        mov  ax,9fffh ;Enviar palabra

        call env_dat  ;pedir datos

        call ban_lo?  ;Preguntar por bandera de L 0 y continuar
                        ;prog hasta activarse su bandera
        mov  dx,dat_ent;Enviar dato a
        in  ax,dx     ;registro ax
        mov  [si+espejo],ax ;Almacenar en mem de PC

        call cambio  ;Bandera de PC validar datos desactivada

        call ban_baj ;Preguntar por bandera de validar datos de
                        ;L 0 y continuar hasta desactivarse
        inc  si       ;Incrementar en dos posiciones el
        inc  si       ;desplaz de mem de espejo
        cmp  si,3904 ;Se almacenó todo el espejo de la hoja ?
        jnz mem_dat  ;No, seguir almacenando

        ret          ;Si, regresar a prog princ

dat_mem endp        ;Fin proc dat_mem
-----

```

```

;Proc preguntar por dato de L O y terminar
;hasta que L O contesta
datos? proc near

    mov ax,09ffff ;Preguntar si hay datos

    call env_dat ;Enviar palabra por pto
                ;de salida y validar

    call ban_lo? ;Preguntar por bandera que
                ;valida dato de L O y
                ;terminar hasta activarse
    ret ;Regresar prog princ

datos? endp ;Fin proc datos?
;-----
;Proc pedir a L O extraiga una hoja,
;desactivar bandera de PC y hacer retardo
pick proc near

    mov ax,0effeh ;Extraer hoja
    call env_dat ;Enviar palabra por pto
                ;de sal y validar
    call cambio ;Bandera de validación de
                ;datos a 1 (desactivada)
    call retardo ;Hacer retardo

    ret ;Regresar prog. princ.

pick endp ;Fin proc. pick
;-----
;Proc pedir datos
pet_dat proc near

    mov ax,0dffff ;Palabra pedir datos a ax

    call env_dat ;Enviar ax a L O

    call cambio ;Bandera validar datos de PC
                ;a 1 (desactivada)
    call retardo ;Hacer retardo

    ret ;Regresar prog princ

pet_dat endp ;Fin proc pet_dat
;-----
;Proc abrir derivación
abr_der proc near

    mov ax,0eff7h ;comando abrir derivación
    call env_dat ;Enviar comando y validar envío

    call cambio ;Bandera validar datos a 1 (desactivada)

    call retardo ;Hacer retardo

    ret ;Regresar prog princ

abr_der endp ;Fin proc abr_der
;-----

```

```

-----
; Rutina para decodificar y enviar mensaje debido a estado
;1 de L O
esdo_1 proc near

```

```

estado1:
    call edo_1    ;Pedir edo 1 a L O y
                ;almacenar en ax
    mov status1,ax;edo 1 en status1
    cmp ax,0ffffh;Edo 1 es ok ?
    jz bien_1    ;Si, ve a bien

    push ax      ;No, escribir ax en stack

    call cambio  ;Bandera de validación de
                ;datos a 1 (desactivada)
    pop ax       ;Leer ax de stack

    call error_1 ;Enviar tipo de error

mas_1:
    call edo_1    ;Pedir edo 1 a L O y
                ;almacenar en ax
    mov cx,status1;Edo 1 anterior en cx
    cmp ax,cx     ;Edo 1 = Edo 1+1 ?
    jz mas_1     ;Si, pedir otro edo 1
    jmp estado1  ;No, comenzar de nuevo

bien_1:
    call cambio  ;Bandera de validación de
                ;datos a 1 (desactivada)
    call retardo ;Hacer retardo

    ret          ;Regresa a prog princ

```

```

esdo_1 endp      ;Fin proc esdo_1
-----

```

```

;Proc pedir edo 2 L O y enviar mensaje
esdo_2 proc near

```

```

estado2:
    call edo_2    ;Pedir edo 2 a L O y
                ;almacenar en ax
    mov status2,ax;edo 2 en status2
    cmp ax,0ffffh;Edo 2 es OK ?
    jz bien_2    ;Si, ve a bien

    push ax      ;No, escribir ax en stack

    call cambio  ;Bandera de validación de
                ;datos a 2 (desactivada)
    pop ax       ;Regresa AX de stack

    call error_2 ;Enviar tipo de error

mas_2:
    call edo_2    ;Pedir edo. 2 a L.O. y
                ;almacenar en AX
    mov cx,status2;Edo. 2 anterior en CX
    cmp ax,cx     ;Edo. 2 = Edo. 2+1 ?
    jz mas_2     ;Si, pedir otro edo. 2
    jmp estado2  ;No, comenzar de nuevo

```

```

bien_2:    call cambio    ;Bandera de validacion de
           ;datos a 2 (desactivada)
           call retardo  ;Hacer retardo

           ret          ;Regresa a prog princ.

esdo_2    endp          ;Fin proc esdo_2
;-----
;Rutina para decodificar y enviar mensaje debido a estado
;3 de Lect. Optica
esdo_3    proc near

```

```

estado3:
call edo_3    ;Pedir edo 3 a L.O. y
              ;almacenar en AX
mov  status3,ax;edo. 3 en status3
and  ax,4000h ;Ermascarar bits
cmp  ax,4000h ;Error en hoja ?
jnz  acabar  ;No, ir a bien_3
mov  ax,status3;Enviar edo. 3 a ax
and  ax,8000h ;Ermascarar bits
cmp  ax,8000h ;Hay error ?
jz   acabar  ;No, termina rutina
mov  ax,status3;Si, Enviar edo. 3 a ax
and  ax,0008h ;Ermascarar bits
cmp  ax,0008h ;Hay error de SUHF
jnz  suhf    ;Enviar mensaje de error

mov  ax,status3;Enviar edo. 3 a AX
and  ax,0004h ;Ermascarar bits
cmp  ax,0004h ;Hay error SUHR
jnz  suhr    ;Enviar mensaje de error
jmp  acabar  ;Termina proc.

```

```

suhf:
lea  dx,suhf_3 ;Si, envia
mov  ah,9h     ;el siguiente mensaje
int  21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de linea

call abr_der  ;Abrir derivacion

jmp  acabar   ;Termina proc.

```

```

suhr:
lea  dx,suhr_3 ;Si, enviar
mov  ah,9h     ;mensaje
int  21h       ;Llamar a DOS

call crlf     ;Enter y avance de linea

call abr_der  ;Abrir derivacion

```

```

acabar:
    call cambio      ;Bandera de validacion de
                   ;datos a 1 (desactivada)
    call retardo     ;Hacer retardo

    ret              ;Regresa a prog princ

edo_3  endp          ;Fin proc esdo_3
;-----
;Proc para pedir edo 1 a L 0 y terminar hasta
;que contesta L 0 con edo 1
edo_1  proc near

    mov ax,0cfffh   ;Palabra pedir edo 1 en ax
    call env_dat    ;Enviar ax a L 0

    call ban_lo7    ;Preguntar por bandera de
                   ;L 0 y continuar hasta activarse
    call rec_dat    ;Meter edo 1 en ax

    ret              ;Regresar a prog princ

edo_1  endp          ;Fin proc edo_1
;-----
;Proc para pedir edo 2 a L 0 y terminar hasta que contesta
;L 0 con edo 2
edo_2  proc near

    mov ax,0bfffh   ;Palabra pedir edo 2 en ax
    call env_dat    ;Enviar ax a L 0

    call ban_lo7    ;Preguntar por bandera de
                   ;L 0 y continuar hasta activarse
    call rec_dat    ;Meter edo 2 en AX

    ret              ;Regresar a prog princ

edo_2  endp          ;Fin proc edo_2
;-----
;Proc para pedir edo 3 a L 0 y terminar hasta que
;contesta L 0 con edo 3
edo_3  proc near

    mov ax,0afffh   ;Palabra pedir edo. 3 en AX
    call env_dat    ;Enviar AX a Lect. Op.

    call ban_lo7    ;Preguntar por bandera de
                   ;L 0 y continuar hasta activarse
    call rec_dat    ;Meter edo. 3 en AX

    ret              ;Regresar a prog princ

edo_3  endp          ;Fin proc edo_3
;-----

```

```
;Proc para decodificar el tipo de error debido a estado 1 de L O
error_1 proc near
```

```
bit_15:
    mov bx,ax      ;Estado 1 en bx
    and ax,8000h  ;Enmascara bits
    cmp ax,8000h  ;Se activo bit 15 ?
    jz bit_14     ;No, ve a siguiente bit

    lea dx,uno_15 ;Si, envia
    mov ah,9h     ;el siguiente mensaje
    int 21h       ;con ayuda de int 21h

    call crlf     ;Enter y avance de linea
```

```
bit_14:
    mov ax,bx     ;Estado 1 original en AX
    and ax,4000h  ;Enmascara bits
    cmp ax,4000h  ;Se activo bit 14 ?
    jz bit_13     ;No, ve a siguiente bit

    lea dx,uno_14 ;Si, envia
    mov ah,9h     ;el siguiente mensaje
    int 21h       ;con ayuda de int 21h

    call crlf     ;Enter y avance de linea
```

```
bit_13:
    mov ax,bx     ;Estado 1 original en AX
    and ax,2000h  ;Enmascara bits
    cmp ax,2000h  ;Se activo bit 13 ?
    jz bit_12     ;No, ve a siguiente bit

    lea dx,uno_13 ;Si, envia
    mov ah,9h     ;el siguiente mensaje
    int 21h       ;con ayuda de int 21h

    call crlf     ;Enter y avance de linea
```

```
bit_12:
    mov ax,bx     ;Estado 1 original en AX
    and ax,1000h  ;Enmascara bits
    cmp ax,1000h  ;Se activo bit 12 ?
    jz bit_11     ;No, ve a siguiente bit

    lea dx,uno_12 ;Si, envia
    mov ah,9h     ;el siguiente mensaje
    int 21h       ;con ayuda de int 21h

    call crlf     ;Enter y avance de linea
```

```

bit_11:
mov ax,bx ;Estado 1 original en AX
and ax,0800h ;Ermascara bits
cmp ax,0800h ;Se activo bit 11 ?
jz bit_10 ;No, ve a siguiente bit

lea dx,uno_11 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de línea

bit_10:
mov ax,bx ;Estado 1 original en AX
and ax,0400h ;Ermascara bits
cmp ax,0400h ;Se activo bit 10 ?
jz bit_9 ;No, ve a siguiente bit

lea dx,uno_10 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de línea

bit_9:
mov ax,bx ;Estado 1 original en AX
and ax,0200h ;Ermascara bits
cmp ax,0200h ;Se activo bit 9 ?
jz bit_8 ;No, ve a siguiente bit

lea dx,uno_9 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de línea

bit_8:
mov ax,bx ;Estado 1 original en AX
and ax,0100h ;Ermascara bits
cmp ax,0100h ;Se activo bit 8 ?
jz bit_7 ;No, ve a siguiente bit

lea dx,uno_8 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de línea

bit_7:
mov ax,bx ;Estado 1 original en AX
and ax,0080h ;Ermascara bits
cmp ax,0080h ;Se activo bit 7 ?
jz bit_6 ;No, ve a siguiente bit

lea dx,uno_7 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de línea

```

```

bit_6:
mov ax,bx      ;Estado 1 original en AX
and ax,0040h  ;Enmascara bits
cmp ax,0040h  ;Se activo bit
jz bit_5      ;No, ve a siguiente bit

lea dx,uno_6  ;Si, envia
mov ah,9h     ;el siguiente mensaje
int 21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de línea

bit_5:
mov ax,bx      ;Estado 1 original en AX
and ax,0020h  ;Enmascara bits
cmp ax,0020h  ;Se activo bit 5
jz bit_4      ;No, ve a siguiente bit

lea dx,uno_5  ;Si, envia
mov ah,9h     ;el siguiente mensaje
int 21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de línea

bit_4:
mov ax,bx      ;Estado 1 original en AX
and ax,0010h  ;Enmascara bits
cmp ax,0010h  ;Se activo bit 4
jz bit_3      ;No, ve a siguiente bit

lea dx,uno_4  ;Si, envia
mov ah,9h     ;el siguiente mensaje
int 21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de línea

bit_3:
mov ax,bx      ;Estado 1 original en AX
and ax,0008h  ;Enmascara bits
cmp ax,0008h  ;Se activo bit
jz bit_2      ;No, ve a siguiente bit

lea dx,uno_3  ;Si, envia
mov ah,9h     ;el siguiente mensaje
int 21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de línea

bit_2:
mov ax,bx      ;Estado 1 original en AX
and ax,0004h  ;Enmascara bits
cmp ax,0004h  ;Se activo bit 2
jz bit_1      ;No, ve a siguiente bit

lea dx,uno_2  ;Si, envia
mov ah,9h     ;el siguiente mensaje
int 21h       ;con ayuda de int 21h

call crlf     ;Enter y avance de línea

```

```

bit_1:
mov ax,bx ;Estado 1 original en AX
and ax,0002h ;Enmascara bits
cmp ax,0002h ;Se activo bit 1
jz bit_0 ;No, ve a siguiente bit

lea dx,uno_1 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de linea

bit_0:
mov ax,bx ;Estado 1 original en AX
and ax,0001h ;Enmascara bits
cmp ax,0001h ;Se activo bit 0
jz fin_1 ;No,termina codificacion
;de error
lea dx,uno_0 ;Si, envia
mov ah,9h ;el siguiente mensaje
int 21h ;con ayuda de int 21h

call crlf ;Enter y avance de linea

fin_1:
ret ;Regresar

error_1 endp ;Fin de proc. error_1
;-----
;Proc. para decodificar el tipo de error debido a estado 1 de L O
error_2 proc near

;bit15
mov bx,ax ;Estado 2 en bx
and ax,8000h ;Enmascarar bits
cmp ax,8000h ;Se activo bit 15 ?
jz bit12 ;No, ve a siguiente bit

lea dx,dos_15 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS

call crlf ;Enter y avance de linea

bit12:
mov ax,bx ;Estado 2 original en ax
and ax,1000h ;Enmascarar bits
cmp ax,1000h ;Se activo bit 12 ?
jz bit11 ;No, ve a siguiente bit

lea dx,dos_12 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS

call crlf ;Enter y avance de linea

```

bit11:

```
mov ax,bx ;Estado 2 original en ax
and ax,0800h ;Enmascarar bits
cmp ax,0800h ;Se activo bit 11 ?
jz bit10 ;No, ve a siguiente bit
```

```
lea dx,dos_11 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS
```

```
call crlf ;Enter y avance de linea
```

bit10:

```
mov ax,bx ;Estado 2 original en ax
and ax,0400h ;Enmascarar bits
cmp ax,0400h ;Se activo bit 10 ?
jz bit9 ;No, ve a siguiente bit
```

```
lea dx,dos_10 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS
```

```
call crlf ;Enter y avance de linea
```

bit9:

```
mov ax,bx ;Estado 2 original en ax
and ax,0200h ;Enmascarar bits
cmp ax,0200h ;Se activo bit 9 ?
jz bit8 ;No, ve a siguiente bit
```

```
lea dx,dos_9 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS
```

```
call crlf ;Enter y avance de linea
```

bit8:

```
mov ax,bx ;Estado 2 original en ax
and ax,0100h ;Enmascarar bits
cmp ax,0100h ;Se activo bit 8 ?
jz bit7 ;No, ve a siguiente bit
```

```
lea dx,dos_8 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS
```

```
call crlf ;Enter y avance de linea
```

bit7:

```
mov ax,bx ;Estado 2 original en ax
and ax,0080h ;Enmascarar bits
cmp ax,0080h ;Se activo bit 7 ?
jz bit6 ;No, ve a siguiente bit
```

```
lea dx,dos_7 ;Si, enviar
mov ah,9h ;mensaje correspondiente
int 21h ;Llamar a DOS
```

```
call crlf ;Enter y avance de linea
```

```

bit6:  mov ax,bx      ;Estado 2 original en ax
        and ax,0040h ;Enmascarar bits
        cmp ax,0040h ;Se activo bit
        jz bit5    ;No, ve a siguiente bit

        lea dx,uno_6 ;Si, enviar
        mov ah,9h    ;mensaje correspondiente
        int 21h     ;Llamar a DOS

        call crlf   ;Enter y avance de linea

```

```

bit5:  mov ax,bx      ;Estado 2 original en ax
        and ax,0020h ;Enmascara bits
        cmp ax,0020h ;Se activo bit 5
        jz bit4    ;No, ve a siguiente bit

        lea dx,dos_5 ;Si, enviar
        mov ah,9h    ;mensaje correspondiente
        int 21h     ;Llamar a DOS

        call crlf   ;Enter y avance de linea

```

```

bit4:  mov ax,bx      ;Estado 2 original en ax
        and ax,0010h ;Enmascarar bits
        cmp ax,0010h ;Se activo bit 4
        jz bit3    ;No, ve a siguiente bit

        lea dx,dos_4 ;Si, envia
        mov ah,9h    ;mensaje correspondiente
        int 21h     ;Llamar a DOS

        call crlf   ;Enter y avance de linea

```

```

bit3:  mov ax,bx      ;Estado 2 original en ax
        and ax,0008h ;Enmascarar bits
        cmp ax,0008h ;Se activo bit
        jz bit2    ;No, ve a siguiente bit

        lea dx,dos_3 ;Si, enviar
        mov ah,9h    ;mensaje correspondiente
        int 21h     ;Llamar a DOS

        call crlf   ;Enter y avance de linea

```

```

bit2:  mov ax,bx      ;Estado 1 original en ax
        and ax,0004h ;Enmascarar bits
        cmp ax,0004h ;Se activo bit 2
        jz bit1    ;No, ve a siguiente bit

        lea dx,dos_2 ;Si, enviar
        mov ah,9h    ;mensaje correspondiente
        int 21h     ;Llamar a DOS

        call crlf   ;Enter y avance de linea

```

```

bit1:  mov ax,bx      ;Estado 1 original en ax
       and ax,0002h ;Ermascarar bits
       cmp ax,0002h ;Se activo bit 1
       jz bit0     ;No, ve a siguiente bit

       lea dx,dos_1 ;SI, enviar
       mov ah,9h    ;mensaje correspondiente
       int 21h     ;Llamar a DOS

       call crlf   ;Enter y avance de linea

bit0:  mov ax,bx      ;Estado 1 original en ax
       and ax,0001h ;Ermascara bits
       cmp ax,0001h ;Se activo bit 0
       jz fin_2    ;No,terminar codificacion
                       ;de error

       lea dx,dos_0 ;SI, enviar
       mov ah,9h    ;mensaje
       int 21h     ;Llamar a DOS

       call crlf   ;Enter y avance de linea

fin_2:  ret        ;Regresar

error_2 endp ;Fin de proc. error_2
;-----
;Proc. programar PPI de salida y de entrada
ppi proc near

;Programar PPI de entrada

       mov al,9bh  ;Byte prog. PPI entrada
       mov dx,ctr_ent;Dir de pto. de ctrl ent.
       out dx,al   ;Envia byte que programa
                       ;al PPI de entrada

;Programar PPI de salida

       mov al,80h  ;Byte prog PPI salida
       mov dx,ctr_sal;Dir de Pto. ctrl. sal.
       out dx,al   ;Saca byte que programa
                       ;al PPI de salida

       ret        ;Regresa a prog princ

ppi endp ;Fin proc. ppi
;-----
;Proc. hacer retardo
retardo proc near

       mov cx,000ah ;Haz Ah veces el ciclo

repite: loop repite ;Ve a "repite" Ah veces

       ret        ;Regresar Prog. princ.

retardo endp ;Fin proc. retardo
;-----

```

```

;Proc. hacer retardo
retraso proc near

        mov cx,0f000h ;Haz F000h veces el ciclo
repit:  loop repit     ;Ve a "repit" FFFFh veces

        ret          ;Regresar a Prog. princ.

retraso endp      ;fin proc. retraso

```

```

;Proc. preguntar bandera de Lect. Op. y continuar
;hasta desactivar
ban_baj proc near

```

```

        mov dx,val_ent;Meter bandera de Lect. Op.
activa:

        in  al,dx     ;en al
        and al,01h    ;Emascarar bits
        cmp al,01h    ;Activa la bandera?
        jnz activa    ;Si, esperar a que se
                    ;desactive

        ret          ;No, regresar prog. princ.

```

```

ban_baj endp      ;fin proc. ban_baj

```

```

;Proc. preguntar por bandera de Lect. Op. y
;continuar hasta activarse
ban_lo? proc near

```

```

        mov dx,val_ent ;Meter a AL el
desact:

        in  al,dx     ;edo. de bandera
        and al,01h    ;emascarar bits
        cmp al,00h    ;Esta activa la bandera?
        jnz desact    ;No, preguntar por ella

        ret          ;Si, regresar prog. princ.

```

```

ban_lo? endp      ;fin proc. ban_lo?

```

```

;Proc. enviar un enter y un salto de linea
crif . proc near

```

```

        mov dl,0dh    ;Enter
        mov ah,02h    ;Funcion de Display
        int 21h       ;Llamar a DOS

```

```

        mov dl,0ah    ;Avanzar linea
        mov ah,02h    ;funcion de Display
        int 21h       ;Llamar a DOS

```

```

        ret          ;Regresar prog. princ.

```

```

crif endp         ;fin proc. crif

```

```
;Proc. enviar datos y validarlos
env_dat proc near
```

```
mov dx,dst_sal;Enviar dato a pto. sal.
out dx,ax ;en 16 bits y
mov al,00h ;activar se#al de
mov dx,val_sal;validacion de
out dx,al ;dato de salida

ret ;Regresar prog. princ.
```

```
env_dat endp ;Fin proc. env_dat
```

```
-----
;Proc. meter una palabra a registro ax
rec_dat proc near
```

```
mov dx,dst_ent;Meter por pto. de ent.
in ax,dx ;una palabra a ax

ret ;Regresar a prog. princ.
```

```
rec_dat endp ;Fin proc. rec_dat
```

```
-----
;Proc. deshabilitar bit 0 pto. c de sal. bit
;para validar datos de sal.
cambio proc near
```

```
mov al,01h ;Levantar bandera
mov dx,val_sal;de salida
out dx,al ;de PC (Desactivada)

ret ;Regresar prog. princ.
```

```
cambio endp ;Fin proc. cambio
```

```
-----
;Proc. probar si hoja leída es de aplicación disponible
aplic_? proc near
```

```
mov apli_no,00;Ceros en apli_no

mov di,00 ;Inicializar contador No. de marcas
mov si,0000 ;Inicializar localización de marcas en mem.

only:
mov ax,[si+marca] ;Enviar a ax # de ovalo a calificar

call renglon ;Encontrar renglon de ovalo y ajustar
;desplaz. de mem. (bx)
call desplaz ;encontrar posición de memoria debido al
;desplaz. del ovalo en el renglon y ajustar
;apuntador de mem. bx

and al,0f0h ;Enmascarar los 4 bits mas significativos
mov ah,00 ;ah=00
cmp ax,0060h ;Aplicación seleccionada?
jg no_apli ;No, ve a notificar
inc di ;Si, incrementar contador de marcas
cmp di,13 ;Se probaron 13 marcas
jnz only ;No, ve a probar otra
jmp yes_ap ;Si, terminar proc.
```

```

no_apli:
mov apli_no,01 ;Mas_hoja=01, la aplicacion no es adecuada
yes_ap:
ret ;Regresar a prog. princ.

aplic_? endp ;Fin proc. aplic_?
-----
;Proc. para encontrar en que renglon se encuentra
;el ovalo a calificar
renglon proc near

inc si ;Aumentar 2 la loc. mem.
inc si
mov cl,72 ;Cargar a cl # de ovalos por renglon
div cl ;Divide No. de ovalos por renglon entre
;el No. de ovalo
cmp al,0 ;Es cero el residuo ?
jz ya ;Si, ve a terminar
;No, ver en que renglon se encuentra el ovalo
push ax ;Guardar en el stack valor de ax
mov ah,00 ;Mover cero a ah
;ax contiene el # de renglon
mov cl,32 ;Enviar 32 a cl
mul cl ;Multiplicar el # de renglon por 32
;para situar loc. de mem. que le corresponde
;al renglon del ovalo
mov bx,ax ;Pasas loc. de mem. del renglon a bx
pop ax ;Regresar # de renglon

ya:
ret ;Regresar

renglon endp ;Fin proc. renglon
-----
;Proc. encontrar posicion de memoria debido
;al desplaz. del ovalo dentro del renglon
desplaz proc near

mov al,ah ;Mover desplaz. del renglon a al
mov ah,00 ;Enviar ceros a ah
mov cl,08 ;Enviar 8 a cl
div cl ;Dividir desplaz. del renglon entre 8
;Residuo de desplaz. en ah
;Cociente de desplaz. en al
cmp ah,6 ;Residuo = 6 ?
jnz cero ;No, preguntar por otro residuo
sub bx,4 ;Inicializar desplaz. de mem.

call loc ;Si, buscar loc. de mem.

jmp final ;Terminar proc.

cero:
cmp ah,00 ;Residuo = 0 ?
jnz dos ;No, preguntar por otro residuo
mov cl,09 ;Si, Mover a ch el cociente mayor+1
sub bx,3 ;Inicializar desplaz. de mem.

call loc ;Buscar loc. de mem.

jmp final ;Terminar proc.

```

```

dos:
    cmp ah,2      ;Es el residuo=2
    jnz cuatro   ;No, preguntar por otro residuo
    sub bx,2      ;Inicializar desplaz. de mem.

    call loc     ;Si, llamar a rutina para buscar
                ;loc. de mem.
    jmp final    ;Terminar proc.

cuatro:
    cmp ah,04    ;Residuo = 4 ?
    jnz final    ;No, terminar proc.
    sub bx,01    ;Si, Inicializar desplaz. de mem.

    call loc     ;Buscar loc. de mem.

final:
    ret         ;Regresar prog. princ.

desplaz endp   ;Fin proc. desplaz.
;-----
;Proc. almacenar dato del ovalo en el reg. al
loc proc near

    sub cl,al    ;Restar 08 a cl
    mov al,04   ;Enviar 04 a al
    mul cl      ;Multiplicar cl por al
    add bx,ax   ;Sumar localidad de renglon mas
                ;desplaz. y enviar a bx
    mov al,[bx] ;Enviar a al contenido del
                ;ovalo pedido

    ret        ;Regresar

loc endp      ;Fin proc. loc
;-----
;Proc. almacenar datos de los ovalos a calificar (folio y calificaciones)
fol_cal proc near

    mov dl,00   ;Inicializar # de ovalos a almacenar
    mov si,0000 ;Inicializar loc.de memoria del # de ovalos

others:
    mov ax,[si+No_ova] ;Enviar a ax el # de ovalo a almacenar
    call renglon      ;Investigar en que renglon se encuentra el
                    ;ovalo y ajustar apuntador de memoria (bx)
    call desplaz      ;Encontrar posición de memoria debido a
                    ;desplaz. del ovalo en el renglon y ajustar bx

    mov [di+info],al ;Enviar info. del ovalo

    inc di           ;Incrementar contador de mem.
    inc dl           ;Incrementar dl
    cmp dl,195      ;Se leyeron 195 ovalos (folio+calif) ?
    jnz others      ;No, ve a leer otro

    ret             ;Si, regresar a prog. princ.

fol_cal endp      ;Fin proc. fol_cal
;-----

```

```

;Proc. almacenar datos de las correcciones
recal proc near

    mov cx,0000 ;Inicializar contador del max. de correcciones

o_corr:
    push cx ;Enviar al stack
    mov di,00 ;Inicializar contador del max. # de ovalos
    ;para renglon donde se va a hacer correccion

oth:
    mov ax,[si+No_ova] ;Enviar a ax # de ovalo a almacenar
    inc ax ;Incrementar # de ovalo

    call renglon ;Encontrar renglon de ovalo y ajustar
    ;apuntador de memoria bx
    call desplaz ;Encontrar posición de mem. debido a
    ;desplaz. del ovalo dentro del renglon
    ;y ajustar apuntador de memoria bx

    mov [di+info],al ;Enviar info. del ovalo a memoria
    inc di ;Incrementar desplaz. de mem.
    inc di ;Incrementar contador de ovalos por renglon
    cmp di,13 ;Se leyeron 13 ovalos ?
    jnz oth ;No, ir a leer otro
    mov dh,00 ;Si, inicializar contador del # de letras

o_cal:
    mov ax,[si+No_ova] ;Enviar a ax # de ovalo a almacenar

    call renglon ;Encontrar renglon de ovalo y ajustar
    ;apuntador de mem. bx
    call desplaz ;Encontrar posición de mem. debido a
    ;desplaz. del ovalo dentro del renglon
    ;y ajustar apuntador de memoria bx

    mov [di+info],al ;Enviar info. del ovalo a mem.
    inc di ;Incrementar desplaz. de mem.
    inc dh ;Incrementar contador # de calificación
    cmp dh,05 ;Ya se almacenaron 5 calificaciones ?
    jnz o_cal ;No, ir a almacenar la siguiente
    pop cx ;Si, regresar contador del max. de correcciones
    inc cx ;Incrementar # de correccion
    cmp cx,0003 ;Se almacenaron 3 correcciones
    jnz o_corr ;No, ir a almacenar la siguiente

    ret ;Si, regresar

recal endp ;Fin proc. recal
;-----
;Proc. procesar aplicacion
aplica proc near

    call nu_folio ;Calificar folio

    call letras ;Calificar evaluacion

    call correc ;Calificar correcciones

    ret ;Regresar a prog. princ.

aplica endp ;Fin proc. aplica
;-----

```

;Proc. obtener folio  
nu\_folio proc near

```
mov  apli_no,01;Inicializar var. existencia de folio
mov  bx,0000  ;Inicializar apuntador de memoria
mov  cl,00    ;Inicializar contador del max. de eventos
mov  ch,00    ;Inicializar contador de columnas del folio
mov  dh,2fh   ;Inicializar valor ASCII de los numeros
mov  di,0000  ;Inicializar desplaz. de memoria de bodega
mov  si,0000  ;Inicializar desplaz. de memoria del
                    ;resultado de aplicacion
sig:  inc  cl    ;Incrementar cl
      cmp  cl,11 ;Se leyeron 10 ovalos?
      jz   vacio ;Si, terminar esta columna
      mov  al,(si+info);No, enviar a al el ovalo leído de Folio
      inc  dh    ;Incrementar valor ASCII del ovalo a procesar
      inc  si    ;Incrementar desplaz. de memoria
      and  al,DF0h ;Enmascarar los bits menos significativos
      mov  ah,00 ;Cero en ah
      cmp  ax,0060h ;El ovalo esta marcado?
      jg   sig   ;No, otro ovalo
      mov  (di+evalua),dh ;Pasara a zona de almacenamiento el valor
                    ;ASCII del # correspondiente
      inc  di    ;Incrementar desplaz. de mem. de resultados
      mov  dh,2fh ;Inicializar valor ASCII de los numeros
      add  bx,10 ;Actualizar desplaz. de mem.
      mov  si,bx ;y pasa a si
      mov  cl,00 ;Inicializar contador del max. de ovalos a leer
      inc  ch    ;Incrementar # de columnas leidas
      cmp  ch,7  ;7 columnas?
      jnz  sig   ;No, ir a evaluar otra
      jmp  si_folio ;Terminar
```

vacio:

```
mov  apli_no,00;Mueve a no_info zeros para avisar que no
                    ;existe folio
```

si\_folio:

```
ret    ;Si, terminar proc.
```

nu\_folio endp ;Fin de proc. nu\_folio

-----  
;Proc. obtener calificaciones MB, B, S, NA, NP para cada alumno  
letras proc near

```
mov  dl,00    ;Inicializar contador de max. # de
                    ;renglones a evaluar
mov  ch,00    ;Inicializar contador del # de calificaciones
ot_cal: inc  ch    ;Incrementar contador del # de calificaciones
      cmp  ch,26 ;Se procesaron 25 calificaciones?
      jz   term  ;Si, terminar
      mov  cl,00 ;No, inicializar contador max. # de
                    ;ovalos (5) por renglon a evaluar
blanco: inc  cl    ;Incrementar cl
      cmp  cl,06 ;Se leyeron 5 ovalos de calificaciones?
      jz   ot_cal ;Si, No hay calificacion y ve a inicializar
                    ;condiciones
```

```

mov al,[si+info] ;No, enviar a al el valor del ovalo
inc si           ;Incrementar desplaz. de memoria
and al,0f0h     ;Enmascarar los 4 bits menos significativos
                ;del ovalo leido
mov ah,00       ;Enviar ceros a ah
cmp ax,0060h    ;El ovalo esta lleno
jg blanco       ;No, procesar siguiente ovalo
add bx,05       ;Actualizar desplaz. de memoria
mov si,bx       ;y enviolo a SI

call equiv      ;Enviar a almacenar las calificaciones correspondientes
jmp ot_cal      ;Ir a procesar otra calificacion

term:

ret             ;Regresar

letras endp    ;fin proc. letras
;
;Proc. almacenar calificaciones respectivas
equiv proc near

    cmp cl,01   ;El contador es 1
    jz MB       ;SI, la calif. es MB
    cmp cl,02   ;No, contador es 2?
    jz B        ;SI, la calif. es B
    cmp cl,03   ;No, contador es 3?
    jz S        ;SI, la calif. es S
    cmp cl,04   ;No, contador es 4?
    jz NA       ;SI, la calif. es NA
    cmp cl,05   ;No, contador es 5?
    jz NP       ;SI, la calif. es NP
    jmp cal_ot  ;Termina proc.

MB:  mov [di+evalua],'M' ;Almacenar M
     inc di              ;Incrementar desplaz. de mem.
     mov [di+evalua],'B' ;Almacenar B
     inc di              ;Incrementar desplaz. de mem.
     jmp cal_ot         ;Terminar proc.

B:   mov [di+evalua],'B' ;Almacenar B
     inc di              ;Incrementar desplaz. de mem.
     jmp cal_ot         ;Terminar proc.

S:   mov [di+evalua],'S' ;Almacenar S
     inc di              ;Incrementar desplaz. de mem.
     jmp cal_ot         ;Terminar proc.

NA:  mov [di+evalua],'N' ;Almacenar N
     inc di              ;Incrementar desplaz. de mem.
     mov [di+evalua],'A' ;Almacenar A
     inc di              ;Incrementar desplaz. de mem.
     jmp cal_ot         ;Terminar prog.

NP:  mov [di+evalua],'N' ;Almacenar N
     inc di              ;Incrementar desplaz. de mem.
     mov [di+evalua],'P' ;Almacenar P
     inc di              ;Incrementar desplaz. de mem.

```

```

cal_ot:      ret          ;Regresar

equiv      endp          ;Fin proc. equiv
;-----
;Proc. procesar correcciones
correc     proc near

mas_co:     mov dh,00    ;Inicializar contador max. # de correcciones

            inc dh       ;Incrementar contador max. # de correcciones
            cmp dh,04    ;Son 3 correcciones?
            jz no_corr   ;Si, terminar proc.
            mov cl,00    ;Inicializar contador max. # de oвалos
                        ;del primer dígito de la correccion
            mov dl,2fh   ;Inicializar valor ASCII de #'s

fol:        inc cl       ;Incrementar contador max. # de oвалos
            cmp cl,04    ;Se leyeron 3 oвалos?
            jz no_corr   ;Si, no hay correcciones

            mov al,(si+info) ;No, enviar a al ovalo leido del # de correccion
            inc dl       ;Incrementar valor ASCII del resultado
            inc si       ;Incrementar desplaz. de memoria
            and al,0fh   ;Enmascarar bits menos significativos
            mov ah,00    ;Enviar a ah cero
            cmp ax,0006h ;El ovalo esta marcado
            jg fol       ;No, procesar otro ovalo

            mov [di+evalua],dl ;Si, enviar a mem. valor ASCII del # correspondiente
            inc di       ;Incrementar desplaz. de mem.
            add bx,03    ;Actualiza desplaz. de mem.
            mov si,bx   ;en si

;Procesar segundo dígito del renglon de correccion

            mov cl,00    ;Inicializar contador del max. # de oвалos
                        ;del segundo dígito de la correccion
            mov dl,2fh   ;Inicializar valor ASCII de #'s

sigui:      inc cl       ;Incrementar contador de max. # de oвалos
            cmp cl,10    ;Se leyeron 10 oвалos?
            jz no_corr   ;Si, no hay correccion
            mov al,[si+info] ;No, enviar a al siguiente ovalo
            inc si       ;Incrementar desplaz. de mem. de espejo de hoja
            inc dl       ;Incrementar valor ASCII del siguiente #
            and al,0fh   ;Enmascarar los cuatro bits menos significativos
            mov ah,00    ;Enviar ceros a ah
            cmp ax,0006h ;Esta lleno el ovalo?
            jg sigui     ;No, procesar el siguiente
            mov [di+evalua],dl ;Si, enviar valor ASCII correspondiente
            inc di       ;Incrementar desplaz. de mem. de aplicacion
            add bx,10    ;Actualizar desplaz. de mem.
            mov si,bx   ;en si
;-----

```

;Procesar calificaciones de correcciones

```
mov cl,00 ;Inicializar contador max. # de ovalos a
           ;evaluar del primer dígito de la correccion
mov ch,00 ;Inicializar contador de max. # de ovalos
           ;a evaluar del segundo dígito de la correccion
ot_cor:   inc ch ;Incrementar contador max. # de ovalos a evaluar
           ;del segundo dígito de la correccion
           cmp ch,6 ;Se procesaron 5 calificaciones?
           jz no_corr ;Si, ir a procesar otra calificacion
```

white:

```
mov al,[si+info] ;No, enviar a al valor del ovalo
inc si ;Incrementar desplaz. de memoria de almacen
inc cl ;Incrementar cl
and al,0f0h ;Enmascarar 4 bits mas significativos de ovalo leído
mov ah,00 ;Enviar zeros a ah
cmp ax,0060h ;Esta lleno el ovalo?
jg white ;No, procesar siguiente ovalo

call equiv ;Enviar almacenar calificaciones correspondientes
add bx,05 ;Actualizar desplaz. de mem.
mov si,bx ;en si

jmp mas_co ;Ir a procesar otra correccion
```

no\_corr: ret ;Regresar

correc endp ;fin proc. correc

-----  
;Enviar retorno y avance de línea cada calif. de hoja  
cr\_lf .proc near

```
mov [di+evalua],0ah ;Enviar retorno
inc di ;Incrementar loc. de mem.
mov [di+evalua],0ah ;Enviar avance de línea
inc di ;Incrementar loc. de mem.

ret ;Regresar
```

cr\_lf endp ;fin proc. cr\_lf

-----  
;Proc para limpiar pantalla

```
mov ah,06h ;Iniciar RAM de color
mov al,00 ;Transferir a ES
mov cx,0000 ;Dir. de inicio
mov dx,2479h ;Carácter a imprimir
mov bh,07h ;Poner atributo normal
int 10h ;Llamar a ROM video
```

;Posicionar cursor en parte media de pantalla

```
mov ah,02 ;Func. posicionar cursor
mov dh,09h ;Region de inicio
mov dl,00 ;Columna de inicio
mov bh,00 ;Página actual
int 10h ;Llamar a ROM video
-----
```

;Proc. enviar letrero # de hojas calificadas

```
mov dx,offset nom_arch ;Envia letrero SU INFORMACION FUE ALMACENADA
mov ah,09h             ;EN EL ARCHIVO:
int 21h

mov dx,offset nombuff+2 ;Enviar nombre del archivo
mov ah,09h             ;Funcion desplegar mensaje
int 21h                ;Llamar a DOS

call crlf              ;Retornar y av. línea
call crlf              ;Retornar y av. línea

mov dx,offset hoj_Cal ;Envia letrero NUMERO DE HOJAS CALIFICADAS:
mov ah,09h             ;Funcion desplegar mensaje
int 21h                ;Llamar a DOS
mov bx,hojas           ;Enviar a BX el numero de hojas calificadas
```

;BX tiene # hexa a convertir y visualizar

-----  
;Proc. para convertir # hexadecimal en BX a  
;un # decimal en pantalla

```
mov cx,10000 ;Dividir entre 10000
call dec_div
mov cx,1000 ;Dividir entre 1000
call dec_div
mov cx,100 ;Dividir entre 100
call dec_div
mov cx,10 ;Dividir entre 10
call dec_div
mov cx,1 ;Dividir entre 1
call dec_div
call crlf ;Retornar y av. línea
call crlf ;Retornar y av. línea
call crlf ;Retornar y av. línea

ret ;Regresar
```

-----  
;Proc. para dividir # hexadecimal en BX entre  
;un # en decimal en cx  
dec\_div proc near

```
mov ax,bx ;# mitad alta
mov dx,0000 ;0 en mitad baja
div cx ;Dividir por cx
mov bx,dx ;Residuo en bx
mov dl,al ;Cociente en dl
```

;Imprimir contenido de dl en pantalla

```
add dl,30h ;Convertir a ASCII
mov ah,02h ;Funcion de desplegar
int 21h ;Llamar a DOS

ret ;Regresar
```

dec\_div endp ;Fin de proc. dec\_div

-----

;Proc para limpiar pantalla

```
mov ah,06h ;Iniciar RAM de color
mov al,00 ;Transferir a ES
mov cx,0000 ;Dir. de inicio
mov dx,2479h ;Caracter a imprimir
mov bh,07h ;Poner atributo normal
int 10h ;Llamar a ROM video
```

;Posicionar cursor en parte media de pantalla

```
mov ah,02 ;Func. posicionar cursor
mov dh,09h ;Renglon de inicio
mov dl,00 ;Columna de inicio
mov bh,00 ;Pagina actual
int 10h ;Llamar a ROM video
```

-----  
;Proc. enviar letrero # de hojas calificadas

```
mov dx,offset nom_arch ;Envia letrero SU INFORMACION FUE ALMACENADA
mov ah,09h ;EN EL ARCHIVO:
int 21h
```

```
mov dx,offset nonbuff+2 ;Enviar nombre del archivo
mov ah,09h ;Funcion desplegar mensaje
int 21h ;Llamar a DOS
```

```
call crlf ;Retornar y av. línea
call crlf ;Retornar y av. línea
```

```
mov dx,offset hoj_Col ;Envia letrero NUMERO DE HOJAS CALIFICADAS:
mov ah,09h ;Funcion desplegar mensaje
int 21h ;Llamar a DOS
```

```
mov bx,hojas ;Enviar a BX el numero de hojas calificadas
```

;BX tiene # hexa a convertir y visualizar

-----  
;Proc. para convertir # hexadecimal en BX a  
;un # decimal en pantalla

```
mov cx,10000 ;Dividir entre 10000
call dec_div
mov cx,1000 ;Dividir entre 1000
call dec_div
mov cx,100 ;Dividir entre 100
call dec_div
mov cx,10 ;Dividir entre 10
call dec_div
mov cx,1 ;Dividir entre 1
call dec_div
```

```
call crlf ;Retornar y av. línea
call crlf ;Retornar y av. línea
call crlf ;Retornar y av. línea
```

```
ret ;Regresar
```

```

;Proc. para dividir # hexadecimal en BX entre
;un # en decimal en cx
dec_div proc near

    mov ax,bx      ;# mitad alta
    mov dx,0000    ;0 en mitad baja
    div cx         ;Dividir por cx
    mov bx,dx      ;Residuo en bx
    mov di,al      ;Cociente en di

;Imprimir contenido de di en pantalla

    add di,30h     ;Convertir a ASCII
    mov ah,02h     ;Funcion de desplegar
    int 21h        ;Llamar a DOS

    ret           ;Regresar

dec_div endp      ;Fin de proc. dec_div
-----

```

## CAPITULO IV. MANUAL DE USUARIO

### A. FUNCIONAMIENTO A TRAVES DE LA PC-XT

#### A.1 DESCRIPCION GENERAL

Una vez culminado el programa de control de la interfaz, se procedió a la adecuación de éste para una fácil manipulación de la lectora óptica a partir de una serie de comandos o instrucciones de fácil manejo que se explicarán posteriormente.

A partir de estos comandos o instrucciones, el operador carga el programa de control, inicializa las actividades de lectura, monitorea en pantalla algunos errores y carga el programa de aplicación.

#### A.2 RUTINA PARA CORRER PROGRAMA DE APLICACION

1. Revisar que la tarjeta de expansión este conectada en la PC-XT y que el cable de interfaz este conectado a la lectora óptica.
2. Encender la PC-XT y la lectora óptica.
3. Colocar en el alimentador de hojas de la lectora óptica las hojas que se leerán.

DESDE LA PC-XT :

4. Estando en C:\>, cambiarse al subdirectorío LECT\_OPT y teclear LECTURA. Cabe mencionar que este archivo es ejecutable y al momento de correrlo aparece una pantalla de presentación con la siguiente información.

- UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
- DIRECCION DE SERVICIOS DE COMPUTO PARA LA ADMINISTRACION ACADEMICA
- 8 APLICACIONES DIFERENTES DE LECTURA OPTICA

5. Elegir la aplicación correspondiente.

NOTA: Este programa únicamente soporta las aplicaciones 5 y 6 correspondientes a las actas de exámenes ordinarios y extraordinarios consecutivamente debido a que el objetivo de este trabajo de tesis fué el diseñar la interfaz de control y controlar la lectora a través de la PC\_XT. Sin embargo se decidió elaborar estas dos aplicaciones para mostrar prácticamente el funcionamiento de la tarjeta de interfaz, dejando a los

especialistas en programación la elaboración de las demás aplicaciones.

6. De acuerdo a lo anterior, si se tecldea un número de uno a ocho, diferente de cinco o seis, el programa envía un mensaje de aplicación no disponible y termina regresando a DOS.

7. Si se selecciona cualquier otra tecla, el programa envía un mensaje de selección inadecuada y permite teclar nuevamente otra opción.

8. Al seleccionar la aplicación cinco o seis, el programa solicita el nombre de un archivo donde almacenará la información de las hojas calificadas. El nombre del archivo no deberá de exceder de ocho caracteres alfanuméricos y si desea colocar una extensión al nombre, ésta no deberá exceder de tres caracteres.

9. Después de haber teclado el nombre del archivo y "ENTER", automáticamente la PC-XT gobierna a la lectora óptica y se inicia la lectura de las hojas.

10. Una vez culminada la lectura de las hojas, el archivo automáticamente se cierra y envía a pantalla el nombre del archivo donde se almacenó la información y el número de hojas leídas, e inmediatamente regresa a DOS.

11. En caso de presentarse algún error en el transcurso de la lectura, el programa es capaz de detectarlo y enviar a pantalla un mensaje con el tipo de error ocurrido y la lectora detiene su motor. De acuerdo al tipo de error, este se corregirá manualmente en la lectora óptica, se encendera el motor de ésta y automáticamente la lectora continuará trabajando.

## **B. CONTROL A TRAVES DE LA PC-XT**

### **B.1 INTRODUCCION**

Como podrá observarse, el programa ejecutable denominado LECTURA instalado en el subdirectorío LECT\_OPT es el programa de control que gobierna a la lectora óptica y es capaz de sensar diferentes tipos de errores, además de tener incluido en dicho programa dos aplicaciones de lectura para comprobar la efectividad del diseño de la tarjeta de expansión o interfaz.

En el momento en que se corre el programa LECTURA automáticamente la PC-XT adquiere el control de la lectora óptica y si se presenta algun error en el proceso de lectura este podrá ser corregido por el operador de acuerdo al mensaje que aparezca en pantalla, posteriormente se rehabilitara el switch de encendido del motor de la lectora y esta continuará su trabajo.

## B.2 PROGRAMAS CON EXTENSION EXE

Existen fundamentalmente dos clases de archivos ejecutables conocidos como archivos com y archivos exe los cuales se cargan en memoria y se ejecutan como programas.

Los programas con extensión exe tienen varias ventajas sobre los programas con extensión com. En lo que se refiere al diseño de aplicaciones, sus características principales son:

- Programas extremadamente largos
- Segmentos múltiples
- Overlays
- Constantes de segmento y dirección distante
- Llamadas largas
- La posibilidad de implementar los programas al modo protegido de MS OS/DOS

Las principales ventajas del formato exe las proporciona la cabecera de archivo que contiene información que permite a un programa hacer referencias directas de dirección de segmento - un requerimiento si el programa está planeado para ser mayor de 64 KB.

La cabecera de archivo también indica a MS-DOS el total de memoria requerida por el programa. Esta información guarda a la memoria no requerida por el programa de ser asignada a éste - una importante consideración si se desea que en el futuro el programa corra eficientemente bajo el modo protegido de MS OS/DOS.

La cabecera de los programas exe también incluye la tabla de apuntadores de relocalización de dirección de segmento directo.

Resumiendo, las estructuras de archivos y de programas con extensión exe permiten una considerable flexibilidad en el diseño de programas, dando al programador la libertad necesaria para producir aplicaciones a gran escala.

## B.3 PROGRAMAS CON EXTENSION COM

Un archivo com es la forma más simple de un programa, solo consta de números binarios que corren el programa, no maneja ninguna otra información adicional y usa muy poca memoria.

Los programas con extensión com son programas simples y pequeños, carecen de información de cabecera. La ausencia de cabecera deja al MS-DOS sin la posibilidad de conocer cuanta memoria requieren los programas en adición al tamaño de la imagen del programa. Debido a esto, MS-DOS debe asignar siempre los bloques más grandes de memoria libre al programa com. Generalmente MS-DOS asigna el bloque completo de memoria libre remanente.

Los programas con extensión com no pueden hacer referencia de dirección a las etiquetas especificadas en las directivas del segmento debido a que carecen de la tabla de apuntadores de

relocalización de dirección de segmento directo.

La estructura del programa com existe en primer lugar para soportar el vasto número de programas CP/M importados a MS-DOS. Estos programas se utilizan comunmente para evitar que se sumen al programa los 512 Bytes o más de la cabecera de información de los programas exe.

La estructura de los programas com tiene la ventaja de que la organización de memoria coloca el PSP dentro del mismo segmento de dirección que el resto del programa. Por tanto es más facil acceder campos dentro del PSP en los programas com.

Resumiendo, la estructura de los archivos y programas com es simple y más restringida para escribir programas que la estructura de los programas exe porque los programadores solo tienen un modelo de memoria simple del cual escoger (el modelo del segmento del programa com). Los archivos de los programas com no contienen los 512 bytes o más de la cabecera inherente a los programas exe, así la estructura de los programas com está bien adaptada a pegeños programas para los cuales los 512 bytes de la cabecera podrían ser al menos el doble del tamaño del archivo.

Analizando las características antes mencionadas y de acuerdo a las necesidades del programa de control, se decidio utilizar un archivo exe para el programa de control de la lectora óptica.

## C O N C L U S I O N E S

El objetivo de este trabajo de tesis se culmina exitosamente al lograr controlar una lectora óptica a partir de una PC-XT.

Cuando se presenta un problema, siempre surgen varias alternativas de solución y el camino a seguir depende en gran medida de los recursos económicos y humanos, así como el tiempo que se dedique a la solución.

En este sentido, la solución propuesta a partir del diseño de una tarjeta de expansión o interfaz para una PC-XT, requirió de bastante tiempo de investigación para decodificar todas las señales proporcionadas por la lectora óptica, recurriendo en ocasiones a los diagramas electrónicos de la lectora, además de realizar pruebas con equipo analizador de estados lógicos, hasta culminar con la conexión de la tarjeta diseñada a la lectora óptica.

Paralelamente a la investigación se diseñó e implementó la tarjeta de interfaz digital.

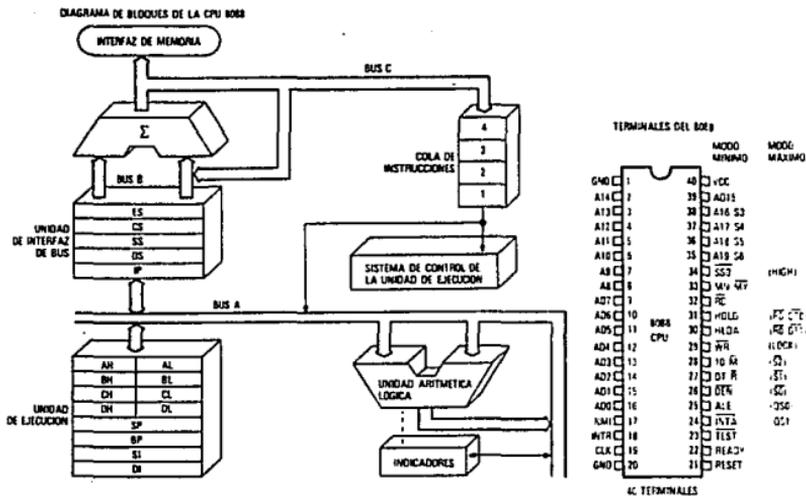
Una vez que se dieron los acontecimientos anteriores, se procedió al desarrollo del programa de control que día a día crecía en tamaño debido a todas las facilidades que debería brindar y que al final cumplió con todas las características consideradas.

A la fecha y debido a las necesidades del servicio de lectura óptica, la Dirección de Servicios de Cómputo para la Administración Académica adquirió dos equipos de lectura óptica con dos computadoras AT 286 como equipo de control, sin embargo aquella Dirección todavía cuenta con papelería de lectura que podrá ser aprovechada utilizando el presente trabajo de tesis por un periodo aproximado de tres años, trabajando los equipos al 80% de su capacidad.

Por tal motivo, el presente trabajo servirá como apoyo directo a la Dirección de Cómputo para la Administración Académica y redundará en un mayor beneficio para todas las dependencias universitarias que requieren los servicios de lectura óptica y así, una vez más, la Universidad Nacional Autónoma de México muestra la confianza depositada en sus egresados para hacerlos partícipes y colaboradores en la solución de problemas actuales que atañen a la comunidad universitaria.

# A P E N D I C E " A "

Diagrama a bloques y patigrama del microprocesador 8088.



## B I B L I O G R A F I A

- BYTE. THE SMALL SYSTEM JOURNAL  
INSIDE THE IBM PC's  
VOL. 12 NUM. 12, 1987
- COLUMBIA DATA PRODUCTS  
MANUAL PARA LA PC-XT
- INTEL  
MICROPROCESSOR AND PERIPHERAL  
HANDBOOK, VOL. II
- LAFORE, ROBERT  
ASSEMBLY LANGUAGE PRIMER FOR  
THE IBM PC & XT
- MANO, MORRIS  
ARQUITECTURA DE COMPUTADORES  
ED. PRENTICE HALL
- MURRAY, WILLIAM  
80386/80286 PROGRAMACION EN  
LENGUAJE ENSAMBLADOR
- RECTOR, RUSELL  
THE 8086 BOOK  
ED. OSBORNE/MC. GRAW HILL
- TEXAS INSTRUMENTS  
TTL DATA BOOK
- WESTINGHOUSE LEARNING CORPORATION  
DOCUMENT SCANNER  
LOGIC DIAGRAMS, VOL.1
- WESTINGHOUSE LEARNING CORPORATION  
DOCUMENT SCANNER  
LOGIC DIAGRAMS, VOL.2
- WESTINGHOUSE LEARNING CORPORATION  
DOCUMENT SCANNER  
LAYOUT SHEETS, VOL 3
- WESTINGHOUSE LEARNING CORPORATION  
LOGIC DIAGRAMS
- YEUNG, BIK  
8086-8088 ASSEMBLY LANGUAGE PROGRAMMING  
JOHN WILEY & SONS