39
24

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"DESARROLLO DE UN SISTEMA LECTOR DE DISCO OPTICO (CD-ROM) PARA UN SISTEMA BASADO EN EL MICROPROCESADOR 68000"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRISISTA

P R E S E N T A N :

JOSE LUIS DAMIAN GOVEA
RICARDO ZALDIVAR GOMEZ

DIRECTOR DE TESIS: M. I. LAURO SANTIAGO CRUZ

MEXICO, D.F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

Pag.

CAPITULO IV.

ANALISIS Y DISEÑO DEL SISTEMA.

CAPITULO V

DESARROLLO DEL SISTEMA Y RESULTADOS

INTRODUCCION

La presente tesis tiene como objetivo Desarrollar un Sistema Lector de Discos Opticos CD-ROM, para un sistema basado en la familia del microprocesador 68000.

La tecnologia de los Discos Compactos (CD) es relativamente nueva. El CD es el nombre que se asigno al disco que se fabrica mediante un proceso de plástico inyectado, al que posteriomente se cubre con una capa de aluminio muy fina. Es un disco de 12 cm de diámetro, de una sola cara que puede almacenar datos digitales en forma de marcas microscópicas las cualés se pueden interpretar por medio de un rayo laser.

Fue creado por Philips y Sony, y originalmente diseñado para almacenar música de alta fidelidad; por esta razón, el estándar con el que se fabrica es un formato que es aceptado en todo el mundo y se conoce como el libro rojo ("The Red Book").

Por la capacidad que el CD tiene para almacenar gran cantidad de datos en un espacio pequeño (aproximadamente 600Mb), ahora es usado como um medio electrónico para almacenar textos o datos.

La aplicación de los CDs a sistemas de cómputo como medios de almacenamiento es muy reciente, dando origen al CD-ROM, el cual es una versión del CD estándar. El nombre de CD-ROM (Compact Disc Read Only Memory) viene del hecho de que los CDs solo se pueden grabar una vez, aunque se pueden leer infinidad de veces, es decir, que la información que contienen no se puede modificar.

El CD-ROM es un producto que se desarrolló utilizando la tecnología para el Compact Disk (CD) de audio digital. El CD contiene unicamente audio digitalizado para ser reproducido utilizando convertidores digital-analógico muy rápidos. En el CD-ROM unicamente se almacenan datos digitales y los convertidores digital-analógico se sustituyen por una interfaz para computadora como la SCSI (Small Computer Standard Interface).

El estándar para la creación de CD-ROMs se especifica en el documento conocido como Libro Amarillo ("The Yellow book Standard").

Las aplicaciones típicas del CD-ROM se refieren al almacenamiento de grandes cantidades de información que no se modifica constantemente, tales como: bases de datos para enciclopedias; registros bibliotecarios (se aumentan pero pocas veces se modifican); registros civiles (defunciones, nacimientos, etc., que quedan registradas y no cambian); etc.

Debido a que este tipo de discos es muy nuevo y cada fabricante utiliza un formato particular para organizar la información en el disco. Actualmente se está tratando de establecer un estándar, basándose en la Propuesta del Grupo High Sierra, el cual está formado por un grupo de vendedores de computadoras, desarrolladores de software e integrantes de sistemas CD-ROM. Una propuesta formal salió en mayo de 1987, por lo que no existe mucha información al respecto; no obstante, se trata de que este trabajo contenga la mayor información posible acerca del tema de CD-ROM para que sirva posteriormente como una fuente de información sobre el tema para todos aquellos que se interesen en él.

El manejo del CD-ROM por medio de la computadora se realiza utilizando herramientas de software y de hardware. Esta última se refiere básicamente al uso de la interfaz SCSI para la comunicación de la computadora con el reproductor de CD-ROM.

Por otro lado, hablar del software que se involucra en el manejo del CD-ROM, es un campo mucho más amplio y en el que se va a hacer énfasis en esta tesis, ya que en este momento se

cuenta con el hardware, pero no con el software para el sistema de computo que vamos a utilizar.

Para desarrollar el sistema lector de CD-ROM para un sistema basado en el microprocesador 68000, es necesario tener la parte correspondiente al hardware. Y por otro lado, con lo que respecta al nivel software se debe tener primero un "DRIVER" o manejador de disco que permita relacionar las señales de control (hardware) con el software que ya soporta el sistema de cómputo (sistema operativo principalmente), y se debe contar también con rutinas generales que permitan accesar la información del disco.

Es bueno mencionar que el CD ROM ya se está utilizando en México.

La UNAM adquirió un equipo CD-ROM para la Dirección General de Bibliotecas, a principios de 1988, que utiliza discos CD-ROM con el Formato High Sierra, y aparte cuenta también con el equipo que se emplea para desarrollar el sistema lector para CD-ROM, haciendo con esto posible que el desarrollo de este trabajo traiga consigo beneficios para la UNAM. Una de las primeras aplicaciones que le están dando a este disco, es el almacenamiento de todas las fichas bibliográficas de tesis y libros que existen en el sistema de bibliotecas de la UNAM; sin embargo, posteriormente se podrán adicionar más fichas bibliográficas, utilizando los conjuntos de Multivolumen del CD-ROM.

Con lo que respecta a esta tesis, ésta se encuentra dividida en 5 capítulos, más las conclusiones, apéndices y bibliografía. A continuación daremos una breve introducción a cada uno de ellos.

En el capítulo uno se proporciona información sobre la familia del microprocesador 68000, en la cual se incluyen sus antecedentes históricos, una descripción física y lógica del microprocesador, así como del sistema de cómputo en general. También se describe al reproductor de CD-ROM, incluyendo a la interfase SCSI que utiliza para conecectarse con la computadora, donde se presentan sus características principales, señales de control y la configuración completa del bus SCSI.

En el capítulo dos se proporcionan las características principales del CD-ROM tanto físicas como de funcionamiento, incluyendo además una tabla de comparación de algunas de estas características del CD-ROM contra otros dispositivos de almacenamiento de información. También incluimos información sobre la forma que se organizan los datos en un disco CD-ROM.

En el capítulo tres se describe el formato High Sierra, y se explica como se organizan los datos en el disco utilizando este formato, tratando de detallar lo más posible la forma en que se manejan los archivos y directorios, además, se describen algunas estructuras importantes del disco, como son: el XAR, la Tabla de Ruta, etc. Por otro lado, también se ofrece la información sobre las características más importantes de la propuesta como son: Archivos Intercalados, Archivos ocultos (hidden), Conjuntos Multivolumen, etc. Todas éstas forman parte del Nivel Tres de la propuesta HSG, por lo cual, proporcionamos en un subcapítulo las capacidades y limitaciones que tiene cada uno de los tres niveles de dicha propuesta.

El capítulo cuatro contiene el análisis y diseño del sistema que queremos desarrollar. Primero abordamos el análisis del sistema, haciendo énfasis en los siguientes puntos: análisis de factibilidad, análisis de alternativas y análisis de requeriminetos tanto de hardware como de software. Posteriormente abordaremos el diseño del sistema, definiendo primero las funciones del sistema y después los distintos procesos del sistema, para finalmente hacer una integración de todos estos procesos.

El capítulo cinco contiene el desarrollo del sistema y los resultados que se obtuvieron del mismo. Proporcionamos información sobre la programación del los distintos procesos del sistema, así como también de las pruebas que se les hicieron y la integración final de dichos procesos.

Por último presentamos los resultados y conclusiones del trabajo, una sección de apéndices y la bibliografía.

CAPITULO I

DESCRIPCION DEL SISTEMA BASADO EN EL up 68000

1.1 INTRODUCCION

Este capitulo describe la operación general del sistema de computo AM-1000 basado en el microprocesador 68000.

La primera parte del capitulo presenta información técnica general, descripción de señales y caracteristicas concernientes con el microprocesador 68000.

La segunda parte describe el funcionamiento general del sistema AM-1000, se provee una descripción técnica general incluyendo un diagrama de bloques simplificado del sistema.

La tercera parte del capitulo describe las caracteristicas y señales de control de la interfaz SCSI (Small Computer System Interface) la cual se emplea para conectar el "drive" del CD-ROM al sistema AM-1000.

Por último se da una descripción de algunas utilerías con las que se cuenta.

1.2 MICROPROCESADOR 68000.

1.2.1 Antecedentes.

El microprocesador MC68000 es el primero de una familia de microprocesadores avanzados de Motorola, para el cual utilizan tecnología VLSI (Very Large Scale Integration). Este microprocesador esta implementado completamente con bus externo de datos de 16 bits y registros de 32 bits. Además cuenta con Set de Instrucciones muy completo y con modos de direccionamiento muy versátiles.

Antes del 68000, Motorola fabricó los microprocesadores de las familias 6500 y 6800 entre otros, de menor capacidad que el 68000, sin contar que fueron hechos con una tecnología diferente y con otra filosofía de operación, lo que hizo que sean incompatibles dichas familias.

Con la construcción del 68000, Motorola perdió continuidad en sus productos en lo que se refiere a operación, ya que las aplicaciones generadas en productos de familias anteriores ya no pueden ser implementadas de forma directa en el 68000. Sin embargo, esto se puede ver también como una ventaja, ya que los nuevos microprocesadores brindan mayor versatilidad y mayores ventajas para aplicaciones en las cuales los otros eran ineficientes.

Motorola sigue con la familia 68000 con productos como el microprocesador 68010, 68020 y 68030; los cuales conservan la misma filosofía de operación que el primero y crecen en capacidad de manejo de datos , velocidad de operación y versatilidad.

1.2.2 Descripción física.

a.) Características y diagramas

Las características del microprocesador 68000 son:

1.-Rango de direccionamiento directo de 16 megabytes
2.-Registros de direcciones y datos de 32 bits
3.-Operaciones con cinco tipos de datos
4.-Memoria mapeada de I/O
5.-14 modos de direccionamiento

Como se muestra en la figura 1.2.2.1 el 68000 ofrece diez y siete registros de 32 bits además de los 32 bits del contador de programa y los 16 bits del registro de status. Los primeros ocho registros (D0-D7) son usados como registros de datos para operaciones con byte (8-bit), word (16-bit) y long word (32-bit).Los registros (A0-A6) pueden ser empleados para operaciones de direccion con words y long words, los diez y siete registros pueden ser usados como registros índice.

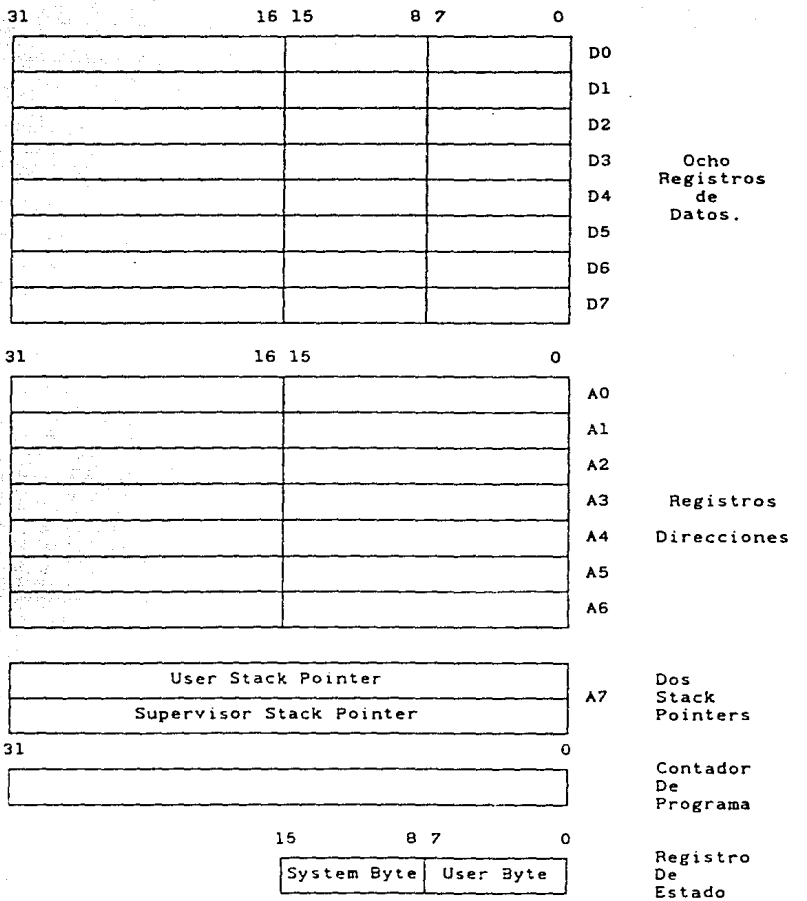

31
16 15
8 7
0
D0
D1
D2
D3
D4
D5
D6
D7
Ocho Registros de Datos.
31
16 15
0
A0
A1
A2
A3
A4
A5
A6
Registros Direcciones
User Stack Pointer
Supervisor Stack Pointer
A7
Dos Stack Pointers
31
0
Contador De Programa
15
8 7
0
System Byte
User Byte
Registro De Estado

Figura 1.2.2.1

La configuración de las terminales para el 68000 se muestran en la figura 1.2.2.2

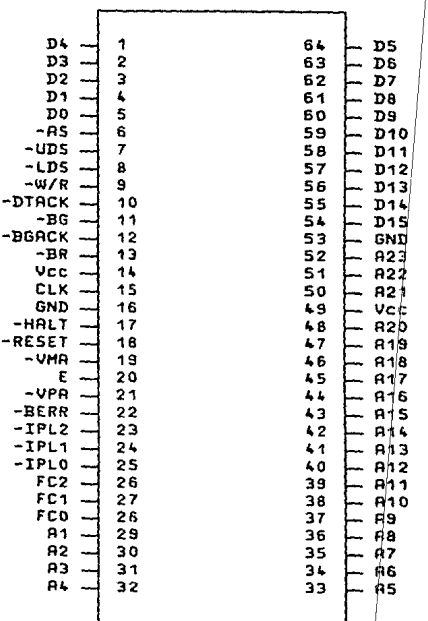


Figura 1.2.2.2

Las señales de entrada y salida pueden ser funcionalmente organizadas en los grupos mostrados en la fig. 1.2.2.3 y los datos son resumidos en la tabla 1.2.2.1. En la sección siguiente se proporciona una descripción breve de cada una de estas señales.

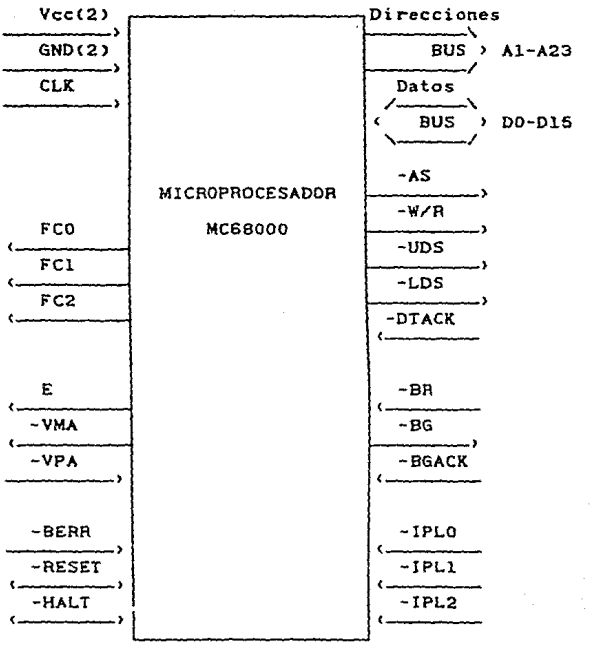

Vcc(2)
GND(2)
CLK
FC0
FC1
FC2
E
-VMA
-VPA
-BERR
-RESET
-HALT
MICROPROCESADOR
MC68000
Direcciones
BUS
A1-A23
Datos
BUS
D0-D15
-AS
-W/R
-UDS
-LDS
-DTACK
-BR
-BG
-BGACK
-IPL0
-IPL1
-IPL2

Figura 1.2.2.3

Nombre de la señal	Nemónico	Input/Output	Estado Activo	Tri estado
Bus de direcciones	A1-A23	Output	alto	si
Bus de Datos	D0-D15	Input/Output	alto	si
Address Strobe	-AS	Output	bajo	si
Read/Write	R/-W	Output	read-alto write-bajo	si
Upper y Lower Data Strobes	-UDS -LDS	Output	bajo	si
Data Transfer Acnowledge	-DTACK	Input	bajo	no
Bus Request	-BR	Input	bajo	no
Bus Grant	-BG	Output	bajo	no
Bus Grant Acknowledge	-BGACK	Input	bajo	no
Interrupt Priority Level	-IPL0 -IPL1 -IPL2	Input	bajo	no
Bus Error	-BERR	Input	bajo	no
Reset	-RESET	Input/Output	bajo	no *
Halt	-HALT	Input/Output	bajo	no *
Enable	E	Output	alto	no
Valid Memory Address	-VMA	Output	bajo	si
Valid Peripheral Address	-VPA	Input	bajo	no
Function Code Output	FC0,FC1 FC2	Output	alto	si
Clock	CLK	Input	alto	no
Power Input	Vcc	Input	-	-
Tierra	GND	Input	-	-

* open drain

Tabla 1.2.2.1

b.) Descripción de las Señales del Microprocesador 68000.

Bus de direcciones. Este bus posee las características siguientes: tiene 23 líneas de dirección, es unidireccional, de tres estados; es capaz de direccionar 16 MB de datos y provee el direccionamiento para las operaciones del bus durante todos los ciclos excepto para el de interrupción, ya que durante este ciclo las líneas de dirección A1,A2 y A3 contienen información sobre el nivel de interrupción que será atendido, por otro lado, las líneas restantes del bus son puestas en "1".

Bus de datos. Este bus tiene las características siguientes : consta de 16 líneas, es bidireccional, de tres estados, y se utiliza para transferencia de información entre el procesador y todos los periféricos, los datos pueden ser enviados en bytes o words. Durante un ciclo de reconocimiento de interrupción, el dispositivo externo suministra el vector de interrupción en las líneas D0-D7.

Control de Bus Asincrono. La transferencia de datos asincronos es manejada utilizando las señales de control siguientes: Address Strobe, Read/Write, Upper y Lower Data Strobes, y Data Transfer Aknowledge, las cuales se explican a continuación.

- Address Strobe (AS). Esta señal indica que existe una dirección valida en el bus de direcciones.

- Read/Write (R/W). Esta señal define la dirección del flujo de información en el bus de datos.

- Upper y Lower Data Strobes (UDS/LDS). Estas señales controlan la información en el bus de datos en combinación con R/W como se muestra en la tabla 1.2.2.2.

- Data Transfer Acknowledge (DTACK). Esta señal indica que la transferencia de datos ha terminado.

Control del Manejo del Bus. Se lleva a cabo utilizando las tres señales siguientes: Bus Request, Bus Grant y Bus Grant Acknowledge. En el sistema que utilizamos estas señales no son utilizadas ya que solo se utilizan en sistemas donde varios dispositivos utilizan el bus de direcciones.

Control de Interrupción (IPL0, IPL1 e IPL2). Estas señales indican en forma codificada el nivel de prioridad del dispositivo que esta solicitando atención. El nivel 7 es el de mayor prioridad, mientras que el nivel 0 indica que ningún dispositivo requiere atención. El bit menos significativo es IPL0 y el más significativo es IPL2.

-UDS	-LDS	R/-W	D6-D15	D0-D7
Alto	Alto	-	Dato no válido	Dato no válido
Bajo	Bajo	Alto	Bits de Datos Válidos 8-15	Bits de Datos Válidos 0-7
Alto	Bajo	Alto	Dato no válido	Bits de Datos Válidos 0-7
Bajo	Alto	Alto	Bits de Datos Válidos 8-15	Dato no válido
Bajo	Bajo	Bajo	Bits de Datos Válidos 8-15	Bits de Datos Válidos 0-7
Alto	Bajo	Bajo	Bits de Datos Válidos 0-7 *	Bits de Datos Válidos 0-7
Bajo	Alto	Bajo	Bits de Datos Válidos 8-15	Bits de Datos Válidos 8-15 *

* estas condiciones son el resultado de la implementación actual y pueden no aparecer en futuros dispositivos.

Tabla 1.2.2.2

Control del Sistema. Es realizado por medio de las siguientes señales: HALT, RESET y BERR; las cuales se utilizan para detener el sistema, inicializarlo o indicarle al procesador que han ocurrido errores en el bus.

- Halt. Cuando esta línea bidireccional es manejada por un dispositivo externo, causará que el procesador se detenga después de completar el ciclo de bus que está ejecutando. Al detenerse el procesador, todas las señales de control son desactivadas y las líneas de tres estados son puestas en el estado de alta impedancia.

En el caso de que el procesador se detenga al ejecutar alguna instrucción (como sucede durante una doble falla en el bus), la línea de HALT la utiliza él mismo para indicarle a los dispositivos externos que se ha detenido.

- **Reset**. Esta señal bidireccional se utiliza para inicializar todo el sistema o solamente los dispositivos externos, el efecto de ésta depende de la forma en que se genera, ya que el primer caso se produce mediante un reset externo y para el segundo caso se necesita que el procesador la genere internamente.

- **Bus Error (BERR)**. Esta señal le indica al procesador que existe algún problema con el ciclo que se está ejecutando; estos problemas pueden ser causados por diferentes motivos como son: dispositivos que no responden, errores al leer el número del vector de interrupción, y otros que dependen de la aplicación. En el sistema que se utiliza los errores de paridad son la única fuente de errores.

Control de Periféricos por el 68000. Estas señales de control se utilizan para la comunicación de los dispositivos periféricos síncronos de la familia 6800 con el 68000 que es asíncrono, dichas señales se explican a continuación:

- **Enable (EN)**. Esta señal se utiliza en forma estándar para habilitar todos los dispositivos periféricos de la familia 6800.

- **Valid Peripheral Address (VPA)**. Es una señal de entrada que indica que el dispositivo o el puerto direccionado es un dispositivo de la familia 6800, por lo cual, la transferencia de información debe ser sincronizada con la señal EN.

- **Valid Memory Address (VMA)**. Esta señal es utilizada por el procesador para indicarle a sus periféricos 68000 que existe una dirección valida en el bus de direcciones y que él está sincronizado al EN. Esta señal unicamente se activa cuando se recibe la señal de VPA la cual indica que el periférico es de la familia 6800.

Estado del Procesador (FC0, FC1, FC2). Estas señales indican en forma codificada el estado y el ciclo que esta siendo ejecutado, como se muestra en la tabla 1.2.2.3. La información seleccionada por éste código, es válida siempre y cuando la señal de AS este activada.

Clock (CLK).La entrada del reloj es una señal compatible TTL la cual es procesada internamente para obtener las señales de reloj requeridas por el procesador. El reloj funciona a una frecuencia constante de 8 Mhz.

FC2	FC1	FC0	Ciclo
Bajo	Bajo	Bajo	(Indefinido, Reservado)
Bajo	Bajo	Alto	Dato de Usuario
Bajo	Alto	Bajo	Programa de Usuario
Bajo	Alto	Alto	(Indefinido, Reservado)
Alto	Bajo	Bajo	(Indefinido, Reservado)
Alto	Bajo	Alto	Dato Supervisor
Alto	Alto	Bajo	Programa Supervisor
Alto	Alto	Alto	Interruption Acknowledge

Tabla 1.2.2.3

c.) Características Eléctricas

En esta sección se presentan algunas especificaciones eléctricas del microprocesador MC68000. Este dispositivo cuenta con una circuitería que protege a las entradas contra daños ocasionados por altos voltajes estáticos o campos eléctricos; sin embargo, se recomienda que las precausiones normales sean tomadas para evitar la aplicación de cualquier voltaje que pueda dañar al mismo.

A continuación se presentan algunas tablas en las que se resumen dichas características eléctricas.

Capacidad Normal

Especificación	Simbolo	Valor	Unidad
Voltaje de alimentación	Vcc	-0.3 a +7.0	V
Voltaje de entrada	Vin	-0.3 a +0.7	V
Rango de Temperatura de Operación: MC68000 MC68000C	T_A	TL a TH 0 a 70 -40 a 85	°C
Temperatura de Almacenamiento	Tstg	-55 a 150	°C

Características Térmicas

Características	Símbolo	Valor	Rating
Resistencia Térmica Cerámica Plástico con Heat Spreader Chip Carrier Tipo B Chip Carrier Tipo C	O_{JA}	 30 30 50 50	°C/W

1.2.3 Descripción Lógica

a.) Conjunto de instrucciones

El microprocesador 68000 cuenta con una gran variedad de instrucciones que pueden ser operadas con bytes, words y longwords. En el Apendice A se muestra una tabla que contiene un resumen de las mismas. Sin embargo, cada sistema en particular cuenta con un conjunto especial de macroinstrucciones, las cuales se forman con un grupo de instrucciones más simples.

1.3 DESCRIPCION GENERAL DEL SISTEMA

1.3.1 Antecedentes.

El sistema con el cual se va a desarrollar el Driver de CDROM es un equipo de cómputo tipo microcomputadora. Es un sistema multiusuario basado en el microprocesador de la familia MC68000.

Dicho sistema tiene aproximadamente 20 años de que se comercializa, en Estados Unidos, donde se encuentra la matriz, en Inglaterra y en toda Latinoamérica.

El sistema lleva como nombre Alpha Micro y opera bajo un sistema operativo propio, el AMOS (Alpha Micro Operating System).

Existen diversas familias dentro de lo que es Alpha Micro. Los sistema capaces de soportar un driver de CDROM, debido a su configuración e interfaces que manejan y demás caracteristicas, son los equipos de la familia AM-1200 en adelante.

1.3.2 Descripción Física

a.) Diagrama de Bloques.

La mayoría de los componentes de la serie AM-1000 funcionan por medio de una estructura de bus interno, tanto de datos como de direcciones , la cual se encuentra bajo el control del CPU.

Un arreglo de estos componentes se muestra en la figura 1.3.2.1 por medio de un diagrama de bloques simplificado.

b.) Descripción del Sistema

A continuación se presenta una descripción general de cada una de las partes que componen al sistema de la serie AM-1000

Unidad Procesadora Central (CPU). El CPU emplea un avanzado procesador 68000 el cual posee las siguientes características:

- Rango de direccionamiento directo de 16MB
- Registros de datos y direcciones de 32 bits
- Funciona con cinco tipos de datos
- Memoria mapeada de I/O
- 14 modos de direccionamiento

Memoria Dinámica RAM. Las series AM-1000 cuenta con un mínimo de 128K de memoria dinámica RAM, y puede ser aumentada en incrementos de 128K o 256K bytes por medio de la tarjeta de expansión de memoria AM-1002. Durante los ciclos de lectura y escritura en memoria, la paridad es checada, y si se presenta un error de paridad este causará un error en el bus el cual es enviado al CPU y esto origina que el led indicador del panel frontal sea activado, para indicar que dicha falla a ocurrido.

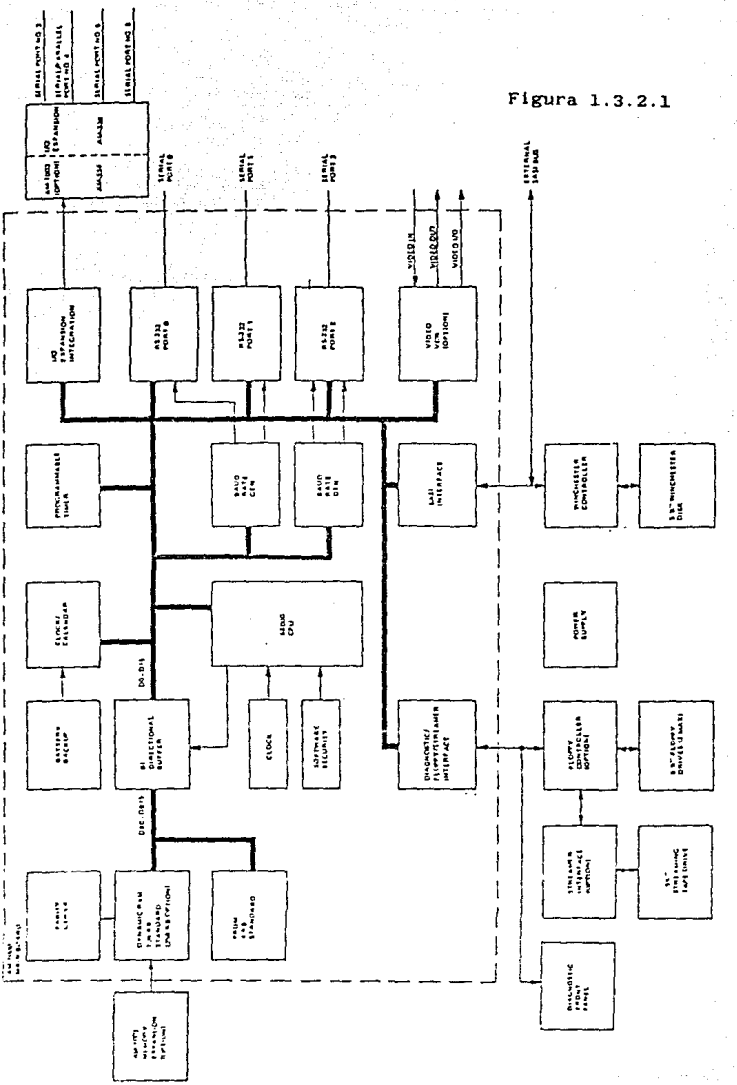
Memoria EPROM. En las series AM-1000 se cuenta con dos circuitos integrados EPROM capaces de almacenar 16KB. Estos EPROMS tienen las siguientes funciones: correr la autoprueba y cargar sistema de disco duro o de videocassete.

Disco. Es un disco magnético tipo winchester de 5 1/4", el cual se puede conectar al procesador a través de las interfaces SASI o SCSI. Acepta discos de diferentes marcas y capacidades.

Reloj de Tiempo Real. El sistema cuenta con un reloj de tiempo real con calendario incluido, para ciertas aplicaciones que lo requieren, así como para el control de archivos. Este reloj es del tipo ininterrumpido. Se cuenta con la opción de programarlo por medio del sistema.

Timer Programable. Está formado por tres contadores que pueden ser programados a ciertos intervalos de tiempo para interrumpir al CPU. Existen comados en el sistema para que el CPU programe y controle al Timer como se requiera.

Figura 1.3.2.1



Interfaz para Puertos Seriales de Entrada y Salida. En forma básica se proporcionan tres puertos seriales RS-232 para la transmisión de datos en forma asincrona entre el sistema y los dispositivos periféricos de entrada y salida (como terminales e impresoras). Consta de tres módulos adaptadores para la interfaz de entrada y salida y de dos módulos generadores de baudios.

Interfaz para VCR (opcional). Esta interfaz se utiliza como medio de respaldo de información utilizando una videocassettera comercial (VCR), para transferir información entre sistemas o para inicializar el sistema por medio de la VCR.

Panel Frontal. Por medio de este panel se despliega información sobre el estado del sistema durante su funcionamiento en autoprueba, o en operación normal indica cuando ha ocurrido un error y de que tipo. Además indica si el sistema esta encendido, esta operando normalmente o si existe algún error de paridad en la memoria.

Bus de interfaz SCSI. El sistema cuenta con una interfaz SCSI (Small Computer Sistem Interface) la cual permite la transferencia de información con el disco interno, tambien cuenta con un conector para poder conectar discos externos para aumentar la capacidad del sistema.

Floppy o Streamer como Subsistema (opcional). Se pueden adicionar al sistema una unidad de Floppy o Streamer, estas opciones se emplean con un propósito similar al de la VCR.

Disco Winchester de Subsistema (opcional). Consiste de un manejador de disco winchester de 5 1/4" y la tarjeta controladora que funciona con la interfaz SASI.

Expansión de Puertos Seriales (opcional). Se cuentan con interfaces con 4 u 8 puertos seriales ó 4 puertos seriales y un puerto paralelo tipo Centronics.

Expansión de Memoria (opcional). Se cuenta con tarjetas de expansión de memoria RAM para llegar hasta 4 MB máximo.

Fuente de Alimentación. El sistema cubre sus necesidades de energía por medio de una fuente de alimentación interna, la cual está diseñada para poder ser utilizada con voltajes de 110/220 VAC. A continuación proporcionamos sus caracteristicas eléctricas:

ESPECIFICACION	REQUERIMIENTOS
Voltaje de Entrada	110 Vac +/- 15% 47-63 Hz ó 220 Vac +/- 15% 47-63 Hz
Alimentación de +5 Vdc	+5 Vdc +/- 5% @ 12 A
Alimentación de +12 Vdc	+12 Vdc +/- 5% @ 4 A
Alimentación de -12 Vdc	-12 Vdc +/- 5% @ 1.5 A
Ruido	50 mV pico a pico

1.3.3 Descripción Lógica.

a.) Software que Soporta.

El sistema está habilitado para soportar una gran cantidad de software, en el que se incluye tanto los lenguajes de programación de alto y bajo nivel como muchos otros programas, algunos de los más importantes se proporcionan a continuación:

Sistema Operativo - AMOS/L, multiusuario, multitareas, tiempo compartido, sistema basado en disco. Incluye macroensamblador de 3 pasos, generador de símbolos, depurador con manejo de símbolos, administración de archivos y otras utilerías. También incluye Alpha BASIC.

Lenguajes y Herramientas de desarrollo - SMC BASIC, FORTRAN 77, COBOL, PASCAL, C, lenguajes de 4a generación, Base de datos UNIFY, utilerías para el soporte de gráficas AMIGOS.

Aplicaciones para Automatización de Oficina - Alpha WRITE, procesador de texto; Alpha CALC, hoja de cálculo; Alpha MAIL, correo electrónico.

Comunicaciones - Redes locales con AlphaNET; AlphaMATE para conexión con computadoras personales PC.

Programas de Aplicación Específicos para Negocios (más de 500 disponibles)

1.4 MANEJO DEL LOS DISCOS.

Lo que se refiere al manejo del disco, es básicamente a la forma en como accesa el sistema operativo AMOS a dicho disco.

Se debe tener al menos un disco físico para el sistema, en el cual se debe localizar: el sistema operativo, los "drivers" para el manejo de los periféricos (incluyendo los discos), las utilerías y todo el software necesario para el uso del sistema y de los paquetes que se proporcionan con él.

Se pueden manejar hasta 4 discos físicos, incluyendo al del sistema. Los discos que no son del sistema se les conoce como "subsistemas" y se utilizan para aumentar la capacidad de almacenamiento de información y ayuda en la organización de la información de las aplicaciones del usuario.

Las rutinas para el manejo de discos, el sistema AMOS las incluye dentro del "Servicio a Disco" (Disk Server) y del "Servicio a Archivos" (File Server).

El sistema operativo es muy general, por lo que necesita de un "driver" o manejador de disco que traduzca las ordenes generadas del sistema a ordenes que el controlador del disco pueda efectuar. es por ésto que se requiere de un "driver" para cada disco físico diferente que se maneje.

1.4.1 Formato de la estructura del disco.

AMOS soporta dos sistemas de archivos distintos: El Tradicional, con 16 bits para direccionar y el tamaño de los archivos limitado a 32 Mb o menos; y el Extendido, el cual usa 32 bit como apuntadores, permitiendo archivos con un tamaño de 2,097,512 Mb. Ambos sistemas de archivos pueden estar presentes al mismo tiempo en el sistema pero en diferentes unidades lógicas, según los requerimientos de un sistema dado.

a.) Formato del Bloque Físico.

El tamaño del bloque lógico para los discos usados dentro de la estructura de archivo de AMOS es de 512 bytes, aunque la mayoría de los drives de disco son formateados por AMOS para que tengan un sector físico de 512 bytes, AMOS proporciona soporte para dispositivos con sectores físicos más pequeños ya que automaticamente agrupa a éstos dentro de un bloque de 512 bytes.

Sin importar el tamaño de sector físico, todas las entradas y salidas del sistema de archivos de AMOS son hechas con bloques de disco de 512 bytes.

b.) Tipos de Bloques de Disco.

El sistema de archivos de AMOS utiliza cinco diferentes tipos de bloques los cuales están clasificados por su uso en el procesado lógico de archivos. Los cinco tipos de bloques son:

A. Bloque de Etiqueta del Disco

B. Bloques de Bitmap

C. Bloques de Directorio

D. Bloques de Datos de Archivos Sequenciales

E. Bloques de Datos de Archivos Contiguos

Los bloques 0,1,2-n (donde n depende del tamaño del disco) contienen información predefinida, sin importar el sistema de archivos. El bloque 0 contiene la etiqueta del disco, el bloque 1 es el primer bloque de directorio del disco, y los bloques del 2-n contienen el bitmap del disco.

El Bloque de Etiqueta del Disco.

Siempre corresponde a el bloque 0 y es usado por los programas LABEL, MOUNT y por algunos otros que necesiten conocer que disco está montado para un driver de disco dado. Este bloque es reservado para la información de identificación del disco, y se cuenta con protecciones para que no se use accidentalmente como bloque de datos.

Este bloque también se utiliza para almacenar las banderas que le indican a AMOS que tipo de directorio (tradicional o extendido) se está usando en la unidad lógica del disco. Sin importar la presencia o ausencia de estas banderas, el formato de este bloque es el mismo para ambos sistemas de archivos.

El Bitmap.

El bitmap consiste de uno o mas bloques de disco que son usados como un mapa de localización de bloques grabados en el disco, el cual siempre inicia en el bloque 2 y se extiende los bloques necesarios para poder cubrir todo el disco. Cada palabra en el bitmap representa el estado de 16 bloques lógicos, usándose un bit para cada bloque. Si el bit es "1" el bloque está en uso y si es "0" está libre. Las últimas dos palabras de cada bloque del bitmap son un doble palabra que proporciona el hash total el cual es usado para mantener la integridad del bitmap durante el proceso. Las palabras que sobran en el último bloque del bitmap no son usadas.

El formato del bitmap es el mismo en los dos tipos de directorio que ya se mencionaron (tradicional y extendido).

Bloques de Directorio.

Los bloques de directorio son usados por ambos sistemas de archivo para definir los archivos de datos que están grabados en el disco, aunque estos sistemas difieren en su implementación, los dos inician su estructura de directorio en el bloque 1 del disco.

Formato Tradicional de los Directorios.

El Formato Tradicional de la Estructura de los Directorios hace uso de dos niveles de jerarquía de directorios, consistiendo de un Master File Directory (MFD) (Directorio Maestro de Archivos) el cual apunta al User File Directory (UFD) (Directorio de Archivos de Usuario) localizado a través de la estructura del disco.

El bloque del MFD siempre inicia en el bloque 1 y forma la base de la estructura de organización de los archivos. Este contiene registros de cuatro palabras por cada PPN (Projet Program Number), el cual es localizado en el disco por un programa. Si se necesitan más registros, bloques adicionales del disco pueden ser ligados al bloque 1 para aumentar el tamaño del MFD.

Los bloques de los UFDs contienen 42 registros de seis palabras cada uno para describir los archivos de usuario en un PPN. La primera palabra de cada bloque de directorio es una palabra de liga al siguiente bloque de directorio, para cubrir las situaciones donde más de 42 archivos están localizados en el área corriente de usuario. El bloque final de directorio tiene un cero en la palabra de liga, indicando que no siguen más bloques de directorio.

Bloques de Directorio en Formato Extendido.

Los bloques de directorio en formato extendido consisten de una jerarquía de directorios multinivel, la raiz de los cuales siempre inicia en el bloque 1 del disco. Sin embargo, por razones de compatibilidad el sistema de archivo es formateado como una estructura de dos niveles igual que el formato tradicional.

Bloques de Datos de Archivos Secuenciales.

Los archivos de datos secuenciales consisten de una serie de bloques de disco ligados juntos por un apuntador de bloque del disco contenido al inicio de cada bloque. El formato tradicional

de los archivos secuenciales tiene dos bytes de liga y 510 bytes de datos en cada bloque. La liga contiene el número del siguiente bloque en el archivo. Un cero indica que éste es el último bloque en el archivo, por lo cual, éste puede tener de 0 a 510 (0 a 508 en el formato extendido) bytes de datos activos en su área de datos, por lo que, uno de los registros del bloque de directorio contiene dicho número. Los archivos secuenciales son procesados normalmente como una larga cadena de bytes desde el inicio al fin.

Bloques de Datos de Archivos Contiguos.

Los bloques de datos de archivos contiguos tienen 512 bytes de datos y no tienen ligas, y deben ser localizados como un grupo de bloques sin bloques intercalados que pertenezcan a otros archivos. Estos deben ser localizados antes de ser empleados, en cambio, los archivos sequenciales son localizados un bloque cada vez como se requiera. Los archivos contiguos permiten procesos de acceso aleatorio, ya que un bloque puede ser localizado por un incremento directo a la base de los bloques.

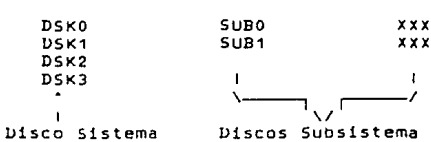
Los archivos contiguos son estructurados en la misma forma bajo el sistema de archivos tradicional y extendido.

1.4.2 Discos Lógicos y Areas.

Cada disco físico es dividido en discos lógicos de igual tamaño. Esta división se puede realizar con comandos del sistema opertativo AMOS. Cada disco lógico se identifica con tres letras y un número que comienza desde cero para cada disco físico. Generalmente las letras del nombre son diferentes para cada disco físico.

Las letras del nombre del disco del sistema son: DSK y es en el disco lógico DSK0 donde se encuentra el sistema operativo.

A continuación presentamos ejemplos de discos lógicos:



Por otro lado cada disco lógico se puede dividir en áreas de trabajo o directorios que el sistema AMOS identifica por medio de 2 números separados por una coma y entre paréntesis cuadrados:

[n,m]

donde el primer número puede identificar un proyecto y el segundo puede identificar al usuario o programador; es por ésto que a las áreas de trabajo se les conoce como PPNs (Project Program Number).

1.5 SISTEMA Y CD-ROM.

1.5.1 Interfaz SCSI.

a.) Características Generales.

El reproductor de CD-ROM siempre opera como un dispositivo "target" (más adelante definiremos este término) y cuenta con 16kB de buffer (opcionalmente se expande a 64 kB), dicho buffer de datos es muy necesario para la transferencia de información a baja y alta velocidad. En operación de lectura, la máxima longitud que puede tener un block de datos es de 2340 bytes. La longitud del block de datos puede ser modificada con un comando. Los procesos de desconexión y reconexión pueden ser asignados cuando los comandos son ejecutados.

La Interfaz SCSI fue establecida por la Engieering Working Group ANSIx3T9.2 de la American National Standards Institute (ANSI) para interfaces específicas entre computadoras anfitrionas y varios periféricos inteligentes. Soporta 27 comandos de operación (incluyendo comandos específicos para el CD-ROM). Las principales características de la SCSI son las siguientes:

+ El dispositivo que envia los comandos se define como Iniciador (Normalmente la computadora anfitriona es el iniciador).

+ El dispositivo que ejecuta los comandos está definido como el dispositivo target (Normalmente el CD-ROM drive opera como dispositivo target).

+ El bus SCSI permite conectar 8 dispositivos SCSI incluyendo al iniciador y los dispositivos target.

+ Ocho fases mostrando el estado del bus SCSI son soportadas.

+ Los identificadores SCSI con direcciones en orden de prioridad en el bus SCSI estan localizados tanto para el iniciador como para los dispositivos target.

+ El estado de comandos está estandarizado.

+ Un cable de 6m máximo terminado con conectores de 50 pins es utilizado para conectar los dispositivos SCSI.

b.) Señales del bus SCSI.

EL bus SCSI consiste de 18 señales, 9 de las cuales son lineas de control y las 9 restantes de datos. En el bus todas las señales son activas bajas. Estas señales son como siguen:

- Señales de datos. (bus de datos)
Es un bus de datos bidireccional que consiste en 8 bits de datos de DB7 a DB0 más un bit de paridad DBp, DB7 es el bit más significativo y DB0 es el bit menos significativo. Este bus es usado con distintos fines, según la operación que se este realizando en el bus:

Arbitration. Se usa para transferir la identificación SCSI para determinar la prioridad para el control del bus. DB7 tiene la mayor prioridad y DB0 la menor.

Selección y Reselección. Se usa para transferir los IDs (identificdor del dispositivo) SCSI entre el iniciador y algún target.

Transferencia de información. Se usa para transferir comandos, datos, estado y mensajes entre el iniciador y un target. DBp es usado como el bit de paridad. El drive de CD-ROM genera un bit de paridad para la salida de datos, y este bit se ignora durante la operación de Arbitration.

+ Bsy (Busy).- Muestra si el bus está ocupado o libre.

+ Sel (Select).- El iniciador utiliza esta señal para seleccionar algún target, y éste reselecciona al iniciador.

+ C/D (Control/Data).- Es usado por el target para determinar si una señal en el bus es de control o de datos. Si C/D = 1 indica que es una señal de control y C/D = 0 indica que es de datos.

+ I/O (Input/Output).- Es usada por el target para saber la dirección de las señales en el bus. La dirección se definida como la ve el iniciador. I/O = 1 indica que las señales van del target al iniciador e I/O = 0 lo contrario.

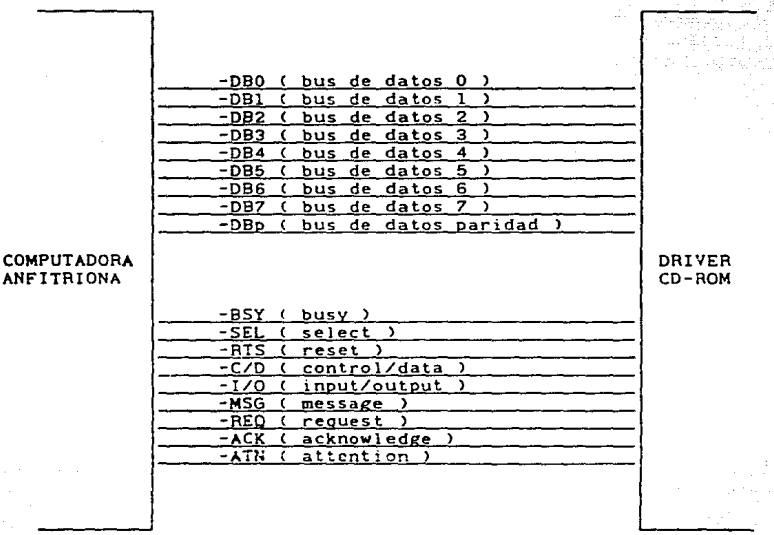
+ MSG (message).- Es usado por el target para reconocer el proceso de mensaje. MSG = 1 indica proceso de mensaje.

+ REQ (Request).- Es usado por el target cuando se ejecuta una trasferencia de datos con REQ/ACK HANDSHAKE.

+ ACK (Acknowladge).- Es controlada por el iniciador cuando se ejecuta una trasferencia de datos con REQ/ACK.

+ ATN (Attention).- Es una señal controlada por el iniciador para mostrar una condición de atención.

+ RTS (Reset).- Es la señal que proporciona la condición de Reset.

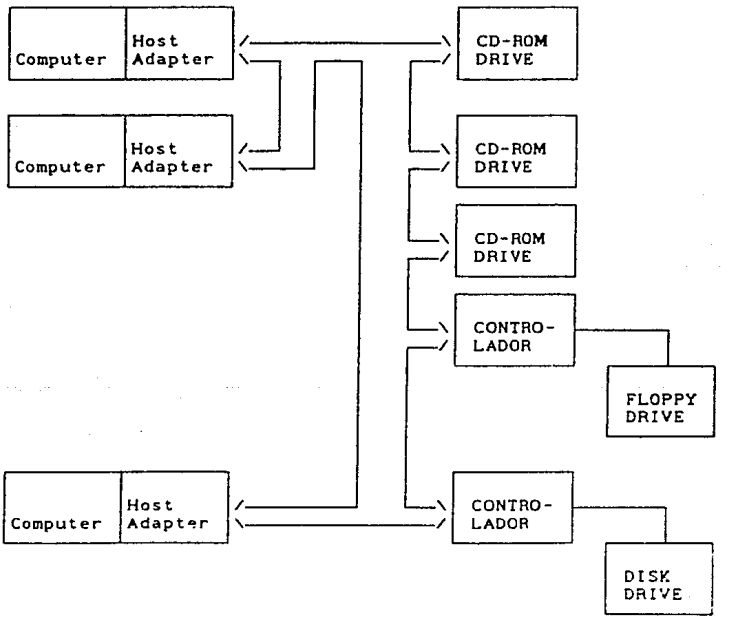


Lineas de señal del bus SCSI.

c.) Especificaciones Físicas.

Ejemplo de la configuración de un sistema.

Un ejemplo de una configuración básica consiste de un iniciador (generalmente la computadora anfitriona) usando el bus SCSI y un target (por ejemplo el driver CD-ROM). Con un sistema que soporte el proceso de ARBITRACION, se pueden conectar hasta 8 dispositivos contando iniciadores, drives CD-ROM y otro tipo de controladores puede ser conectados al bus con el cable que se proporciona con una longitud máxima de 6m y terminados con conectores de 50 pins. En el siguiente diagrama, el drive CD-ROM está integrado por un controlador SCSI y la unidad drive CD-ROM.



Ejemplo de la Configuración de un Sistema con el Bus SCSI.

d.) Características Eléctricas.

La conexión de las señales de la interfaz entre los dispositivos son de tipo daisy chain y son terminadas con resistores en ambos extremos. Hay tres tipos de señales: las que son manejadas por el iniciador, por el target y por ambos al mismo tiempo.

Las señales que manejan los dispositivos SCSI tiene las siguientes características:

VERDADERO (TRUE): VOL = 0.0 - 0.4 V DC
IOL = 48 mA Min. (0.5 V DC)
FALSO (FALSE): VOH = 2.5 - 5.25 V DC

En el driver CD-ROM, el CI-7438 es usado como el circuito driver.

1	G	-DB(0)	26
2	G	-DB(1)	27
3	G	-DB(2)	28
4	G	-DB(3)	29
5	G	-DB(4)	30
6	G	-DB(5)	31
7	G	-DB(6)	32
8	G	-DB(7)	33
9	G	-DB(P)	34
10	G	G	35
11	G	G	36
12	G	G	37
13	OPEN	TERMPWR	38
14	G	G	39
15	G	G	40
16	G	-ATN	41
17	G	G	42
18	G	-BSY	43
19	G	-ACK	44
20	G	-RST	45
21	G	-MSG	46
22	G	-SEL	47
23	G	-C/D	48
24	G	-REQ	49
25	G	-I/O	50

Arreglo de Pins en el conector.

Las señales recibidas por los dispositivos SCSI deben tener las siguientes caracteristicas:

VERDADERO (TRUE):	VOL = 0.0 - 0.8 V DC
	IIL = -0.4mA Max. (0.4 V DC)
FALSO (FALSE):	VIH = 2.0 - 5.25 V DC
HISTERESIS:	0.2 V DC Min.

En el CD-ROM, el CI-LS240 es usado como el circuito de recepción. Cada línea de señal debe ser terminada con 220 ohms (lado de +5V) y 330 ohms (lado de tierra), en ambos extremos de los dispositivos SCSI.

e.) Fases o Condiciones del Bus SCSI.

El bus SCSI siempre se encuentra en alguna de las fases o condiciones cuya descripción se da a continuación:

+ Fase de Bus Libre (Bus Free Phase). Indica que el bus SCSI no está siendo utilizado por ningún dispositivo. En esta fase solo interviene las línas RTS, BSY y SEL, durante esta fase todas las señales del bus son inactivas.

+ Fase de Arbitrio (Arbitrion Phase). En esta fase se determina cual de los dispositivos SCSI tendrá el control del bus, pudiendo ser el iniciador o el target. Es una fase opcional, la cual debe ser realizada si hay varios iniciadores conectados o si el sistema utiliza la fase de RESELECCION. Esta fase puede ser habilitada en el CD-ROM por medio de un switch.

+ Fase de Selección (Selectio Phase). Esta fase es usada por el Inicador para seleccionar un Target. Durante esta fase la señal de I/O es puesta en "0" para identificar la fase de Reselección. El iniciador comienza su operación en esta fase si no se está usando la fase de Arbitrio

+ Fase de Reselección (Reselección Phase).En esta fase se permite a un target que este desconectado de algún iniciador, seleccionar a un iniciador. En esta fase la señal I/O es puesta en "1" para diferenciarla de la fase de Selección.

+ Fase de Transferencia de Información.

Esta fase permite la transferencia de datos, comandos, estado y mensajes por el bus de datos. Los tipos y direcciones de la información que va a ser transferida está determinada por tres señales (C/D, I/O y MSG) que son controladas por el target como se muestra en la siguiente tabla. Además podemos observar que esta fase varia según el tipo de información que se maneje, generando con esto otras nuevas fases que son:

- Fase de Comando.
- Fase de Entrada de Datos.
- Fase de Salida de Datos.
- Fase de Estado.
- Fase de Recepción Mensaje (REC. DE MSG).
- Fase de Transmisión de Mensaje (TRS DE MSG).

Signal			Fase	Dirección de Transferencia
MSG	C/D	I/O		
0	0	0	Salida Datos	I ——› T
0	0	1	Entrada Datos	I ‹—— T
0	1	0	Comando	I ——› T
0	1	1	Estado	I ‹—— T
1	0	0	Sin usar	
1	0	1	Sin usar	
1	1	0	REC. DE MSG	I ——› T
1	1	1	TRS. DE MSG	I ‹—— T

• Condición de Atención. Esta condición indica que el iniciador tiene un mensaje para un target. Esta condición puede ser creada por el iniciador poniendo la señal ATN a "1", excepto durante las fases de Arbitrio y Bus Libre. Cuando se transmiten más de dos bytes de datos, el iniciador debe mantener esta señal en "1".

• Condición de Reset. La condición de Reset tiene prioridad sobre todas las fases o condiciones mencionadas, y cuando se da un reset a los dispositivos SCSI, el sistema comienza a operar con la Fase de Bus Libre. Esta condición puede ser creada por cualquier dispositivo SCSI con solo poner la señal RTS a "1" por más de 25 us.

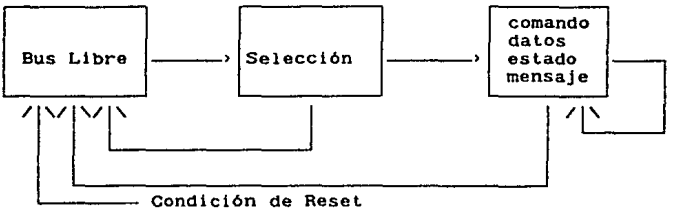
El CD-ROM al detectar esta condición ejecuta un "HARD RESET ", el cual tiene los siguientes efectos:

- Interrumpe todos los comandos que están siendo procesados.

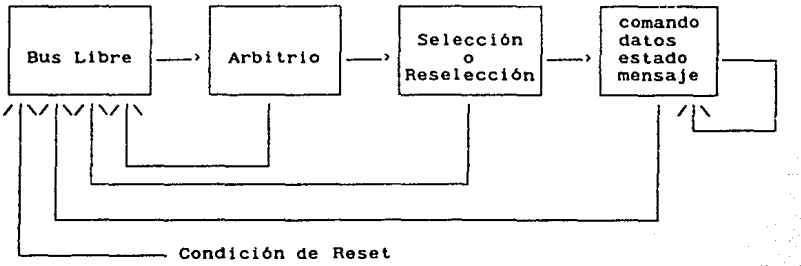
- Libera todos las operaciones o todas las peticiones pendientes.

- Borra el modo especificado y pone el modo inicial.

+ Secuencia de Fases en el bus SCSI. Para realizar todas sus operaciones el Bus SCSI debe seguir una secuencia de fases definida, a continuación mostramos las secuencias de fases para un sistema que cuente con la fase de ARBITRIO y para uno que no la tenga.



Sistema sin Fase de Arbitrio.



Sistema con Fase de Arbitrio.

1.5.2 UTILERIA DEL SISTEMA PARA EL CD-ROM.

Alpha Micro, como es un sistema con independencia de dispositivos, envia con el sistema, una serie de programas que permiten la creacion de Drivers para diferentes discos.

Alpha Micro ha creado un programa para crear Drivers para CD-ROM, se denomina "FIXCD.LIT", dicho programa permite crear driver para discos CD-ROM con formato tradicional y con formato extendido, simulando que el disco es un disco común y corriente de los que maneja AMOS. Simula los sectores que maneja el sistema operativo y las áreas de trabajo.

Al igual que el FIXCD, en el software que se proporciona para el manejo del CD-ROM, viene ya un manejador de CD-ROM denominado "CDRDVR.DVR".

CAPITULO II

CARACTERISTICAS DEL CD-ROM

2.1 INTRODUCCION

La tecnología del CD-ROM (Compact Disc Read Only Memory) se deriva de la tecnología del CD para audio, emplea el mismo mecanismo de drive y el mismo proceso de manufactura del disco. Debido a su gran relación, el desarrollo del disco y del reproductor del CD-ROM se han beneficiado directamente de los avances tecnológicos y reducciones de costo asociados con el rápido crecimiento de la industria de audio del CD.

El formato del disco del CD-ROM es estándar; en comparación con el CD de audio, los discos son del mismo tamaño y sus datos están arreglados exactamente en la misma forma. Esta estructura en común es generalmente conocida como el formato físico.

Teniendo el mismo formato físico se asegura que el CD-ROM que un fabricante hace puede ser leído en cualquier reproductor de otro fabricante.

El formato físico del CD-ROM fué creado por Philips y Sony, los cuales desarrollaron la tecnología del Compact Disc (CD), dicha tecnología se describe en un documento conocido como el Libro Amarillo.

El Libro Amarillo define el tamaño de las marcas microscópicas que representan bits de datos, así como también, que éstas sean arregladas en una espiral continua y que la misma sea dividida en sectores de aproximadamente 2000 bytes de largo y que a su vez cada uno de éstos sean subdivididos en campos.

Además de un formato físico el disco debe contar con un formato lógico para especificar como está organizada la información en el disco. El formato lógico (también llamado formato de archivo) define la organización, tamaño, localización de los archivos en el disco CD-ROM, estructura del directorio para todos los archivos en el disco, y el número de discos y aplicaciones incluídas.

El Libro Amarillo no define el formato lógico del disco, por lo cual éste era determinado por el que desarrollaba el sistema operativo, dando por resultado tantos formatos de archivo como sistemas operativos existieran. Para eliminar este problema, un grupo de compañías interesadas en los discos ópticos de solo-lectura, se unieron recientemente para proponer un formato lógico estándar para el CD-ROM. Este formato ha llegado a ser conocido como High Sierra Proposal (HSP).

2.2 CARACTERISTICAS FISICAS.

2.2.1 Control del Disco, CAV y CLV

La mayoría de los discos magnéticos estan organizados en tracks concéntricos y en sectores , cada uno de los tracks se divide en un número específico de sectores. Por el contrario, el disco del CD-ROM arregla los sectores en una espiral continua. Los diferentes formatos relacionan la forma en la que gira el disco cuando graba o leé. Un disco magnético usualmente gira a razón constante, o a Velocidad Angular Constante (CAV); sin embargo, un disco CD-ROM gira a una razón variable, es decir, más rápido cuando leé los sectores internos y más lento cuando leé los sectores externos, debido a que gira a una Velocidad Lineal Constante (CLV).

a) Velocidad Angular Constante (CAV)

La mayoría de los discos magnéticos y algunos discos de video usan formatos CAV. Un disco CAV gira a una razón constante pero los tracks externos e internos pasan el mecanismo de lectura a diferentes velocidades, los tracks externos viajan más rápido que los tracks internos.

La capacidad de almacenamiento del disco CAV depende de la cantidad de datos que pueden ser almacenados en el radio interno. La figura 2.2.1.1 muestra como están organizados los sectores en el disco CAV, ilustrando las densidades de datos. Como se puede ver los sectores internos son cortos, y sus datos están dénsamente empacados. En el radio externo, los datos són almacenados en sectores más largos y a una densidad lineal menor.

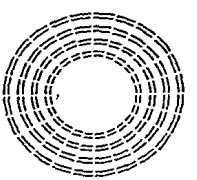


Figura 2.2.1.1. Círculos concéntricos de un disco CAV. Los sectores externos son más largos que los sectores internos, sin embargo, almacenan la misma cantidad de datos.

b) Velocidad Lineal Constante (CLV)

Al contrario de los discos magnéticos, el disco del CD-ROM gira a una velocidad lineal constante. Un disco CLV rota de tal modo que los sectores de datos pasan la cabeza lectora a una velocidad constante. El drive del CD-ROM mantiene una velocidad lineal constante cambiando la velocidad de rotación del disco conforme la cabeza se mueve a lo largo del track.

Como se vé en la figura 2.2.1.2 el formato CLV del disco CD-ROM dá por resultado que los sectores tengan la misma longitud. Este tipo de formato incrementa la cantidad de datos que el disco puede almacenar ya que permite que tengan más sectores en el radio externo, donde la circunferencia es mas larga.

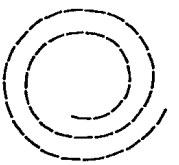


figura 2.2.1.2. Formato espiral de los discos CLV. Todos los sectores son del mismo tamaño.

Aunque el formato CLV es ideal para almacenar grandes cantidades de datos, no lo es para recuperar datos, ya que la larga espiral que el track hace dificulta encontrar sectores individuales. Además cada movimiento de cabeza para encontrar un sector particular debe ser acompañado por un proceso mecánico de acelerar o desacelerar el disco. Esta desventaja influye para un tiempo de acceso de datos relativamente grande para el CD-ROM en comparación con otros dispositivos de almacenamiento de datos.

2.2.2 DIRECCIONAMIENTO

Un disco CLV requiére un esquema diferente de direccionamiento, que lo haga más familiar al esquema de trakcs orientados de los discos magnéticos CAV. Aunque un sistema operativo puede encontrar un sector en la mayoría de los discos magnéticos, usando solo el track y el número del sector, éste necesita saber los minutos y segundos del tiempo de la "reproducción" lineal y el número del sector para encontrar un sector en el CD-ROM. (ver figura 2.2.2.1)

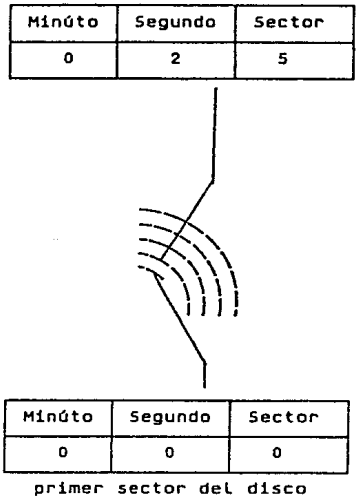


Figura 2.2.2.1 El esquema de direccionamiento físico del CD-ROM viene de la herencia del de CD para audio. Un disco puede almacenar 60 "minutos" de datos, cada minúto es dividido en 60 "segundos". y cada "segundo" de datos es dividido en 75 sectores. Entonces, como un disco almacena 270,000 sectores (60 minutos x 60 segundos x 75 sectores), y como cada sector contiene 2kB de datos, no incluyendo los códigos de sincronía, encabezado, y códigos de detección y corrección de errores (EDC/ECC), un CD-ROM tiene una capacidad de 540,000 kB.

Una aplicación raramente se involucra con la dirección física del CD-ROM, ya que maneja los datos como un grupo de archivos, cada uno de los cuales tiene asociado un nombre. Cuando una aplicación llama a un archivo en particular, el sistema operativo la toma y convierte la visión lógica que la aplicación tiene del disco, en una visión física encontrando así la localización física del archivo en el disco CD-ROM.

2.3 ORGANIZACION DEL DISCO

La superficie del disco del CD-ROM, vista bajo un poderoso microscopio, es una espiral con marcas microscópicas, estas marcas son depresiones en la superficie reflejante del disco llamadas pits las cuales no reflejan el rayo laser empleado para la lectura. El area reflejante entre dos pits se le conoce como land, un uno binario representa la transición de un pit a un land y de un land a un pit, y dos o mas ceros representan la distancia entre las transiciones. El laser y la circuiteria del reproductor del disco, ademas de convertir estas marcas en bits de datos, las procesa y las manda a la computadora divididas en sectores lógicos. Cada sector contiene 2352 bytes de información, de los cuales 2048 bytes son datos de usuario. El sector es dividido como se muestra a continuación:

Datos de Sincronización	12 bytes
Datos del Encabezado	4 bytes
Datos del Usuario	2048 bytes
Código de Detección de Error (EDC)	4 bytes
Espacio sin usar	8 bytes
Código de Corrección de Error (ECC)	276 bytes

2.3.1 Datos.

A. Datos de sincronización. Los sectores no se separan físicamente, en su lugar solo se marca el inicio de cada sector con una secuencia de 12 bytes llamada Código de Sincronía. El drive del CD-ROM no pierde de vista su localización en el disco, haciendo las cuentas del número de bytes que ha pasado y por medio de la búsqueda de información de sincronía. Este usa ambos métodos debido a que los datos de sincronía no son suficientes para establecer los límites entre sectores, ya que la misma secuencia de bytes de sincronía pueden casualmente existir en los datos del usuario.

B. Datos del Encabezado. Los cuatro bytes de encabezado identifican al sector, los primeros tres bytes contienen su dirección, la cual emplea el drive para localizar el sector. El último byte en la secuencia es llamado el byte de modo, éste indica la presencia de los datos de corrección de error. El modo 1 incluye corrección de error y detección de información. El modo 2 no incluye los 280 bytes de código de error y este espacio puede ser empleado para grabar datos adicionales del usuario.

C. **Datos del Usuario.** Aquí es donde el usuario puede grabar información, ya sean textos, gráficas, sonido, o video. El proceso premaestro (éste proceso se aborda en el siguiente apartado) divide los archivos de datos en bloks de 2048 bytes, calcula el código de corrección y detección de error, y adiciona información de encabezado y sincronía.

D. **Espacio sin usar.** Este espacio no es usado en el CD-ROM, pero ha sido movido y redefinido en el estándar CDI (Compact Disc Interactive) el cual permite una presentación interactiva de video, audio, texto y datos.

2.3.2 Códigos.

A. **Código de Detección de Error (EDC).** El EDC es un valor de 16 bits generado de los valores de sincronía, encabezado, y datos de usuario, el cual emplea el driver o la computadora para determinar cuando los datos del usuario leídos del disco son diferentes de los datos originales. El EDC es calculado cuando el disco se encuentra en el proceso premaestro.

B. **Código de corrección de Error (ECC).** Los bytes en el campo de corrección de error son empleados para corregir los datos del usuario cuando un error ha sido detectado. Un algoritmo sofisticado, basado en el código "Reed-Solomon", es usado para corregir datos erróneos.

2.4 CORRECCION DE ERROR

La densidad de datos en el CD es muy grande, aproximadamente 100 millones de bits por centímetro cuadrado. Con densidades tan altas, aún un defecto microscópico puede causar cientos de errores. Para aliviar estos errores el estándar CD-ROM incluye un sistema de corrección de error para asegurar la integridad de datos entregados desde el disco.

En el reproductor del CD-ROM existe el hardware necesario en la detección de errores, el cual compara el valor del código de detección de error con los datos conforme estos pasan en su camino a la computadora. Si el decodificador detecta un error, este le dá la señal al sistema de la computadora. El sistema de la computadora usa los códigos de corrección de error grabados a lo largo de los datos de usuario para regenerar los bits "malos" o bits perdidos.

Los reproductores de audio para CD generalmente tienen su mejor promedio de un error en cada 100 millones de bits (10^{-8}). Para el audio en CD este bajo promedio es aceptable, debido a que el reproductor de audio del CD extrapola por bits perdidos y el oído tiende a compensar las imperfecciones momentáneas del sonido que causan los bits incorregibles.

En el CD-ROM sin embargo, aún un error de 1 bit puede afectar significativamente la integridad de la información.

2.5 CARACTERISTICAS DE FUNCIONAMIENTO

En la tabla 2.5.1 se muestran las características de funcionamiento del CD-ROM y el de otros medios de almacenamiento.

MEDIO DE ALMACENAMIENTO	Disco Winchester Pequeño	ROM Optica Grande	Disco Floppy	Cinta Magnética	Disco Winchester Grande	CD-ROM
Costo del medio (US $)	N/A	15-30	1-5	10-20	N/A	10-20
Costo del drive	500-3,000	7,000-100,000	200-1,500	3,000-15,000	10,000-150,000	500-2,500
Capacidad (en MB)	5-50	1,000-4,000	0.36-1.20	30-300	50-4,000	550-680
Volumen del medio (sec.)	5.25	12.00	5.25	10.50	14.00	4.72
Tiempo de acceso (sec.)	0.03-0.30	0.03-0.40	0.03-0.05	1-40	0.01-0.08	0.40-1
Densidad (bits/in.)	15,000	35,000	10,000	6,250	15,000	35,000
Velocidad (KB/sec.)	625	300	31	500	2,500	150

Tabla 2.5.1. Características de desarrollo del CD-ROM y otros medios de almacenamiento

Debido a las características mecánicas del drive del CD-ROM, el tiempo de acceso de un borde del disco al otro es lento. Por lo tanto es importante para un sistema de recuperación evitar el acceso innecesario. La razón de transferencia de datos, sin embargo, es respetable en relación con la de los otros medios, y solo llega a importar cuando se transfieren grandes cantidades de información como son gráficas de alta resolución, animación o material audiovisual.

2.5.1 Tiempo de Acceso

Para leer un sector, la cabeza se mueve a su colocación aproximada y luego sigue la espiral hasta que encuentra el sector apropiado. Para leer de nuevo el mismo sector, el drive debe saltar atrás de la espiral y esperar a que el sector sea encontrado de nuevo. Como la velocidad del disco en el radio externo (200 rpm) es menor que la mitad de la velocidad en los radios internos (530 rpm), leer las espirales externas toma más tiempo del que se toma para leer las espirales internas. Por lo tanto, el tiempo de acceso del CD-ROM es tambien función del radio donde ocurre la lectura.

El término tiempo de acceso es la medida del tiempo requerida para localizar cierta información especifica en el disco. Este término es frecuentemente mal empleado. Su definición más aceptada incluye tres eventos:

- Tiempo de búsqueda. El tiempo que se toma en posicionar la cabeza en el radio deseado.

- Settling time. El tiempo que la cabeza toma para colocarse en la posición de la localización deseada.

- Retraso de Rotación. El tiempo que el disco debe rotar para obtener el sector deseado bajo la cabeza de lectura.

El tiempo de búsqueda en los CD-ROMs varía en un rango desde 2000 milisegundos (ms) para distancias cortas hasta 800 ms para distancias largas. El tiempo de búsqueda de un buen disco Winchester de 5 1/4 pulgadas puede ser aproximadamente de 30 ms o menos. El settling time es más modesto, y con los CD-ROMs depende casi completamente de la habilidad del drive en posicionar exactamente la cabeza lectora la primera vez. El rango del retraso de rotación para CD-ROMs va desde 60 a 150 ms, comparado con 8.3 ms para los winchesters que rotan a 3600 rpm. De modo que el tiempo promedio de acceso para el drive del CD-ROM debe ser desde 400 ms hasta 1 seg.

2.5.2 Razón de Transferencia

La razón de transferencia se refiere a la velocidad con la cual la computadora anfitriona leé los datos del disco una vez que estos son encontrados. La razón de transferencia no incluye el tiempo de acceso, ya que solo se considera la velocidad a la cual los datos se mueven desde el disco a la computadora anfitriona.

La razón de transferencia del CD-ROM se encuentra entre la del disco de floppy (31kB/seg) y la del Winchester (625 kB/seg). El Libro Amarillo fija la máxima razón de transferencia en 176.40 kB/seg. Esta razón se calcula para el tiempo en el que se leen y transfieren los 2352 bytes de un sector, los cuales incluyen: los bytes de sincronia, de encabezado, los datos de usuario y los códigos de corrección y detección de errores. La razón más citada de 150 kB/seg solo considera el tiempo para transferir los datos del usuario, con cantidades de 2048 bytes por sector.

Los datos erróneos pueden hacer más lenta la razón total de transferencia de datos. Si el EDC detecta errores el hardware detiene la transferencia de datos e inicializa el proceso de corrección de error. Una vez que los datos han sido de nuevo corregidos, el drive comenzará a leer los datos de nuevo.

2.6 CARACTERISTICAS LOGICAS

Para asegurar que los CD-ROMs puedan distribuir información a un amplio rango de usuarios se requiere, además de un formato fisico, un formato lógico. El formato lógico, o formato de archivo, permite a las aplicaciones ver el CD-ROM como una conjunto de archivos.

Viendo el disco como un grupo de archivos, la aplicación puede encontrar los datos llamando al archivo por su nombre, como por ejemplo FILE1.DOC. El Sistema de Archvios del sistema operativo busca el nombre del archivo en el directorio del disco y emplea información asociada para localizar el archivo. Sin este proceso de alto nivel del disco, la aplicación tendria que saber la localización física de cada sector relacionada con el archivo. En lugar de esto, todo lo que tiene que saber es el nombre del archivo, el Sistema de Archivos hace el resto.

2.6.1 Formato Lógico

El formato lógico está dividido en dos estructuras distintas el Volumen de la Tabla de Contenido (VTOC) y la estructura del directorio. La primera contiene información referente al disco, incluyendo la localización del directorio del disco. Cuando el Sistema de Archivos empieza a leer el disco, este leé el VTOC antes que cualquier otra cosa. La segunda especifica la localizacion exacta de los archivos en el disco.

2.7 PROCESO DE PRODUCCION DE UN DISCO CD-ROM

El proceso de producción de un disco CD-ROM involucra tres procesos basicos: Premaestro, Maestro y Copiado.

A continuación se da una breve explicación de cada uno de éstos.

2.7.1 Premaestro

Una vez que todos los datos y códigos de control que serán almacenados en un disco óptico son grabados en una cinta y organizados en bloques de datos de 2048 bytes, se puede proseguir a aplicar el proceso de premaestro a dicha cinta la cual se conoce como premastering tape. El término Premaestro se refiere al proceso en el cual los bytes de corrección y detección de error definidos en la especificación del CD-ROM, son calculados y añadidos a cada bloque de 2048 bytes. Tambien son añadidos 12 bytes de sincronia, 3 bytes de dirección y un byte de modo especificado en el CD-ROM estandar.

2.7.2 Maestro

El término Maestro se refiere al proceso en el cual se fabrica el disco que será utilizado para obtener las copias que se necesiten posteriormente. El primer paso en el proceso de manufactura es el de producir el disco maestro, el cual sera usado en el proceso de copiado. El laser hace las marcas que contienen los datos (pits y lands) en una superficie fotoresistiva, comenzando en el track central y moviéndose hacia afuera en un arreglo espiral. Después de terminado este proceso la maquinaria para el copiado hace un stamper(una placa de metal que contiene la imagen negativa de un disco maestro), a partir del cual, se produce la imagen positiva del disco.

2.7.3 Copiado

Los compact discs son hechos de un plástico de policarbonato, la mayoria de los procesos de manufactura hacen la replica del disco empleando algunos métodos de inyección de molde. La resina del policarbonato es calentada y vaciada en los moldes que forman los discos, una vez que ésta se enfria, el stamper imprime la muestra de datos en el plástico y el disco es puesto en una cámara de vacio, donde se le añade una capa reflectiva de aluminio . Finalmente, la capa reflectiva es revestida con laca protectora.

Se tienen diferentes técnicas para hacer las copias de los discos, pero la mayoria de las plantas que trabajan actualmente en ésto emplean el proceso de inyección de molde.

CAPITULO III

DESCRIPCION DEL FORMATO DEL GRUPO HIGH SIERRA

3.1 INTRODUCCION.

Desde 1985 varios fabricantes de discos ópticos (CD-ROM) han estado tratando de establecer un estándar para localizar directorios y archivos sobre un disco CD-ROM.

La propuesta del Grupo High Sierra (HSG) es una creación de un grupo de vendedores y desarrolladores de sistemas CD-ROM. Esta propuesta inicial es importante porque es un formato común que empiezan a usar algunas compañías importantes como Microware, Microsoft, etc.

La información que presentamos es una síntesis del documento escrito por dos miembros del HSG, por lo que se procuró que ésta no contradiga a la propuesta original "Trabajo escrito sobre el Proceso de Información--Volumen y la Estructura de Archivos de un Disco Optico Compacto de Solo Lectura" (Working Paper for Information Processing--Volume and File Structure of Compact Read-Only Optical Discs for Information Interchange).

Un documento muy importante en la construcción de los discos ópticos es el llamado Libro Amarillo (Yellow Book), ya que dicho documento contiene las especificaciones con las que se fabrican los discos, una de las cuales es el tamaño físico de los sectores en los que se divide el disco, sin embargo, dicho documento no es del dominio público, por lo que la propuesta del HSG maneja al disco por medio de un Formato Lógico.

Para simplificar el diseño del Formato Lógico para el CD-ROM se divide el trabajo en dos pasos: (1) se crea un conjunto de estructuras que proporcionen información acerca de todo el disco, y (2) se elaboran estructuras que describan y localicen los archivos en el disco.

En las secciones siguientes se describen las características principales de la estructura del disco que define la propuesta HSG; iniciando con los sectores lógicos, bloques lógicos, archivos y posteriormente hablaremos sobre las estructuras que describen a todo el disco.

3.2 DEFINICION DEL CONCEPTO FISICO Y LOGICO.

3.2.1 Sector Físico y Lógico.

Un sector físico consta de 2336 bytes, no contando los bytes utilizados para la información del encabezado, 288 bytes son usados para corrección de errores y los 2048 bytes (2KB) restantes pueden contener datos del usuario, este campo de datos de 2 KB es lo que la Propuesta HSG considera como un "Sector Lógico"; y si el tamaño de un sector físico es agrandado en futuras versiones del Libro Amarillo, también se contempla agrandar el tamaño de los sectores lógicos

Un número identifica a cada sector lógico y es conocido como LSN (Logical Sector Number), el primer sector habilitado para el usuario en el CD-ROM tiene la dirección física 00:02:00 y le corresponde el LSN=0.

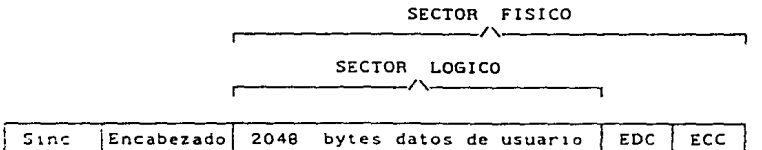


Fig. 3.2.1.1

3.2.2 Bloque Lógico.

Un sector lógico puede dividirse en varios Bloques Lógicos para proporcionar un direccionamiento más preciso. Diferentes CD-ROMs pueden tener varios tamaños de bloques lógicos, los cuales pueden ser de 512, 1024 ó 2048 bytes, sin que ocurra nunca que un bloque lógico sea mayor que un sector lógico.

A cada bloque lógico se le asigna un número conocido como LBN (Logical Block Number). El primer bloque lógico (LBN=0) se encuentra localizado dentro del primer sector lógico (LSN=0). La propuesta HSG recomienda que la información en el disco se direccione en términos de LBNs.

3.3 ARCHIVOS

La propuesta HSG proporciona reglas para identificar archivos y un número de mecanismos para encontrarlos sin especificar que contenido pueden tener los mismos (texto, video, etc.).

Un archivo está asociado con un conjunto de bloques lógicos, los cuales usualmente son subsecuentes, y forman lo que se llama un **Extent**. La propuesta HSG permite tener archivos con varios extents separados, sin embargo, no todos los sistemas operativos permiten multiples extents en el CD-ROM.

3.3.1 Identificadores de Archivos.

Cada archivo tiene que ser diferenciado mediante un identificador (File Identifier), formado por un nombre, una extensión y el número de versión, estas tres cosas son opcionales, sin embargo, para que el identificador sea valido debe tener nombre o extensión. La forma del identificador puede ser alguna de las siguientes:

- Nombre.Extensión;Versión
- .Extensión;Versión
- Nombre;Version
- Nombre.Extensión
- Nombre
- .Extensión

Se pueden utilizar en el identificador los siguientes caracteres:

Números	0-9
Letras	A-Z (solo mayúsculas)
Signos	_

Notese que el nombre y la extensión están separados por un punto y la extensión se separa de la versión por un punto y coma. El máximo de caracteres empleados en un identificador es de 31 incluyendo el punto y el punto y coma, aunque esto varía según diferentes niveles de trabajo en la propuesta HSG, el nivel más bajo tiene 8 caracteres para el nombre del archivo, 3 para la extensión y no permite número de versión, garantizando así la compatibilidad con las PCs.

3.4 DIRECTORIOS.

La propuesta HSG permite la localización de subdirectorios que parten de un directorio raíz o de otro subdirectorio, sin embargo, por las limitaciones de algunos sistemas operativos se limitan a 8 los niveles jerárquicos que pueden tener los subdirectorios.

3.4.1 Estructura de un Archivo Directorio.

Un directorio es similar a un archivo de usuario, sin embargo, la estructura del archivo directorio está definida en forma especial en la propuesta HSG. Los archivos directorios están formados por registros de longitud variable(33 bytes más el número de bytes del identificador), los cuales contienen campos para definir las características del archivo, como son: su identificador, su longitud en bytes, el LBN de su primer bloque lógico en el extent, y otra información requerida para abrirlo y usarlo. Cada extent de archivo (no cada archivo) tiene un diferente registro de directorio.

Cuando un archivo tiene varios extents, el orden de los registros de directorio para ese archivo, permite ordenar logicamente los datos en él, en otras palabras, el primer registro del directorio contiene el primer extent, en el cual se localiza el inicio del archivo y el último registro del directorio contiene el último extent, en el que se encuentra el final del mismo: como se ilustra en la figura 3.4.1.1.

Información adicional acerca de un archivo se puede grabar en lo que se conoce como **XAR** (Extended Attribute Record), su contenido lo abordaremos posteriormente. Por ahora solo mencionaremos que se encuentra localizado inmediatamente antes del archivo en el disco: y el LBN grabado en el directorio para la localización del archivo, apunta al inicio del XAR, y debido a que el registro en el directorio contiene su longitud, el sistema puede facilmente saltarlo y posicionarse al inicio de los datos en el archivo.

Como otros archivos, los subdirectorios son localizados utilizando un registro en el directorio raíz. Por lo cual, la propuesta HSG no define la localización física de los directorios en el disco, pudiendose así localizar según las necesidades de cada aplicación específica. Por ejemplo, localizando el directorio cerca de los archivos que con más frecuencia se accesan se puede minimizar el tiempo de acceso.

La longitud de los registros de directorio es variable, pero no pueden rebasar los límites de un sector lógico. Un sector lógico completo es la cantidad conveniente de datos que se debe leer porque así lo permite el buffer del sistema, y con esto la propuesta HSG asegura que cada buffer contenga parte de un directorio, el cual contiene únicamente registros completos de directorio y nunca fragmentos.

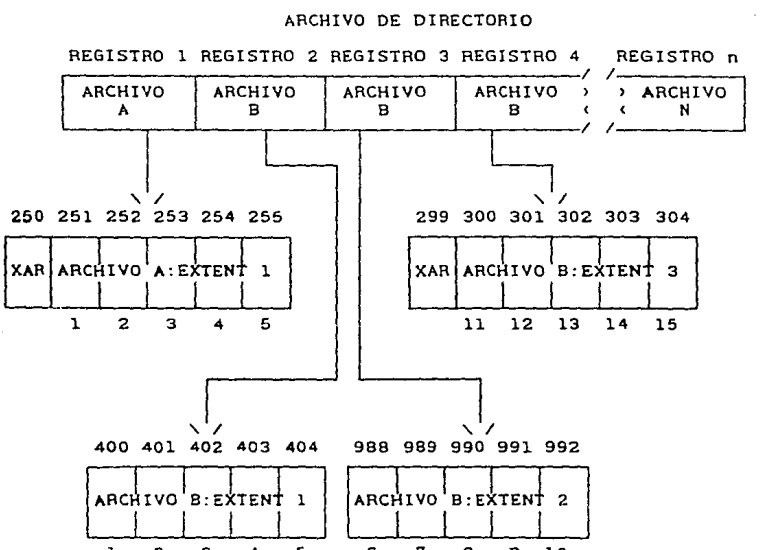


Fig. 3.4.1.1 Organización de los archivos en el disco.

3.4.2 Tabla de Trayectoria.

Cuando los directorios tiene una estructura jerárquica, suele ocurrir que en algunas ocasiones el proceso de búsqueda de un archivo sea muy largo debido a que éste se encuentra muy anidado en el directorio, por lo cual, la propuesta HSG proporciona una forma más rápida para encontrarlo, usando lo que se conoce como "Tabla de Trayectoria" o tabla de ruta (Path Table); ésta contiene el índice de todos los directorios. La propuesta establece que la tabla de trayectoria contenga el LBN de cada subdirectorio, proporcionando acceso directo a un subdirectorio a través de ella. Ahora, si la tabla de trayectoria se carga completa en RAM, una simple búsqueda puede accesar cualquier subdirectorio en el disco.

3.4.3 Funcionamiento de los Directorios.

La tabla de trayectoria únicamente garantiza el acceso al primer sector de directorio, y como un directorio muy largo puede contener miles de archivos y ocupar muchos sectores del disco, lo cual consume mucho tiempo en lectura y búsqueda por parte del sistema para poder encontrar un archivo dentro del directorio, se hace necesario que el usuario distribuya sus archivos en una gran cantidad de subdirectorios, limitando el número de archivos que contenga un subdirectorio a cerca de 40. Los registros de directorio para 40 archivos con nombres de longitud promedio ocupan solo un sector, y si todos los subdirectorios limitan su cantidad de registros a un solo sector, el índice de la tabla de trayectoria proporcionará acceso directo al registro de directorio de un archivo, revelando inmediatamente su localización.

3.5 ATRIBUTOS AMPLIADOS DE LOS REGISTROS (XAR).

Los atributos ampliados de los registros permiten almacenar información adicional acerca de archivos individuales, aumentando y complementando el registro de directorio. Además al hacer ésto se perfecciona significativamente la presentación, y por otro lado, si los registros de directorio son pequeños, cabran más dentro de un bloque de la estructura del directorio, pudiendo así tener información sobre más archivos con una simple búsqueda y lectura del disco. La propuesta HSG permite registros de directorio pequeños para almacenar únicamente la información usada con más frecuencia acerca de cada archivo, y cualquier otra información va en el XAR.

El sistema operativo o la aplicación pueden usar el XAR para guardar cosas como: la estructura del registro asociado al archivo, la creación, modificación, expiración, y datos efectivos del archivo: o información específica de una aplicación especial. Como el uso del XAR es opcional, un campo (Byte Position(BP) 2) en el registro de directorio de cada extent del archivo indica si el XAR está presente. Si una aplicación no necesita de la información en el XAR, el registro simplemente es omitido, y BP 2 es puesto a cero.

Si un archivo tiene multiples extents, cada uno de ellos puede tener su propio XAR, sin embargo, si la información en los XARs llega a tener conflictos, puede ocasionarle problemas al sistema. No obstante, para prevenir esta posible dificultad la propuesta HSG establece que el último extent de un archivo sea el que contenga el XAR válido para todos ellos, ya que si la información en el XAR del último extent es diferente a la de los primeros, estos XARs son ignorados, lo cual es muy bueno para ciertas aplicaciones; por ejemplo, la actualización de archivos.

3.5.1 Localización del XAR.

Cuando una aplicación usa el XAR, éste se pone al inicio del archivo y el valor del campo Longitud del XAR (XAR Length) (BP 2) en el registro de directorio indica cuantos bloques lógicos ocupa. Los datos actuales del archivo inician en el bloque lógico que sigue al XAR.

Los directorios por ser simples archivos, también pueden tener XARs, y cuando los tienen, el campo en la tabla de trayectoria etiquetado como Longitud del XAR (BP 5) indica cuantos bloques lógicos ocupa el XAR, y en donde inician los registros de directorio.

3.5.2 Contenido Estándar de los XARs.

El XAR almacena información acerca del archivo, la que puede ser: el nivel de control de acceso al archivo, sus datos , la estructura de los registros dentro del mismo: y aparte, un campo reservado para uso del sistema, un registro completo del directorio, y un campo reservado para que lo utilice una aplicación especifica. Ademas, estos campos pueden ser empleados tanto por el sistema como por la aplicación que así lo requiera.

1. Nivel de Control de Acceso al Archivo. Los primeros tres campos en el XAR proporcionan información sobre el nivel de control de acceso al archivo estos son como sigue:

1. **Código de Identificación del Propietario.** Este código identifica al propietario del archivo. Es un número de 16 bits que cuando se usa en conjunción con el campo de permiso habilita a un sistema operativo como UNIX o VMS para determinar si el usuario que quiere accesar el archivo tiene el permiso apropiado para hacerlo.

2. **Código de Identificación de Grupo.** Este código especifica el identificador de grupo para el grupo de usuario al cual el propietario del archivo pertenece. El sistema operativo verifica otra vez el permiso apropiado y permite o niega el acceso a miembros de otros grupos.

3. **Permisos.** Este campo especifica las condiciones bajo las cuales se les permitirá a los usuarios el acceso al archivo. (sistema, propietario, grupo y en general)

   El formato de los discos hechos con las especificaciones de la propuesta HSG es muy conocido y probablemente sea implementado en una gran variedad de hardware y de sistemas operativos, algunos de los cuales no tienen control de acceso a los archivos, en tales condiciones no se podrá utilizar la opción de permiso, ya que de lo contrario se presentará la situación, de que usuarios no autorizados tengan a su alcance archivos protegidos.

II. **Datos del Archivo.** El XAR también permite almacenar los datos de cuando el archivo fue creado, modificado, expira o será efectivo. Muchas aplicaciones y sistemas operativos usan estos datos para proporcionar información acerca del uso o estado del archivo. Por ejemplo, si el archivo contiene información que será efectiva en un futuro, la aplicación le indicará al usuario que esa información no es exacta.

III. **Registros.** La parte siguiente del XAR almacena información sobre la estructura de los registros de un archivo. La propuesta HSG soporta dos formatos de registro: longitud fija y longitud variable. El campo del Formato de Registro BP 75 indica si la estructura del registro está definida como longitud fija o variable; un valor de 2 ó 3 en este campo indica el uso de registros con longitud variable, pero distingue entre dos posibles formas de ordenar los bytes, en el campo de la palabra de control del registro, asociado con cada registro de longitud variable en el archivo. Si el valor del formato del registro es 2, entonces, el valor almacenado en la palabra de control del registro es escrita con el bit menos significativo (LSB) primero, y

si el valor es 3, entonces, está escrita con el bit más siginificativo (MSB) primero. A menos que se diga otra cosa, la aplicación grabará todos los archivos como un flujo de bytes que no involucra estructura de registro.

El campo de Atributos del Registro (BP 76) especifica cual es la forma de control de caracteres que será usada para realizar el despliegue de los registros de un archivo. El despliegue se refiere principalmente a una salida, como una pantalla o una hoja impresa, que finalmente muestra el archivo. Un valor de 0 (cero) en este campo indica que una alimentación de linea precede a cada registro en el archivo y un caracter de retorno de carro seguirá a cada registro al ser desplegado. Un valor de 1 indica que el primer byte debe ser interpretado como el caracter de retorno de carro FORTRAN especificado en ISO 1539. Finalmente, un valor de 2 especifica que la aplicación conoce los caracteres de control de forma en el registro y los procesará de acuerdo a ellos.

El campo final de la porción de registros de un XAR es la palabra de control (BP 77-80),la cual contiene un número binario de 16 bits que es interpretado básandose en el valor de BP 75. Cuando el valor en BP 75 es cero, el valor en BP 77-80 debe ser también cero. Esto indica que el archivo no tiene una estructura de registro definida en la propuesta HSG. Un valor de 1 en BP 75 indica que el valor en BP 77-80 determina la longitud de los registros de longitud fija. Un valor de 2 ó 3 en BP 75 establece que el valor en BP 77-80 es la longitud máxima de un registro en el archivo.

IV. Uso del XAR por el Sistema. Un tercer uso del XAR es para almacenar información específica del sistema. El campo identificador del sistema (BP 81-112) contiene el nombre del sistema que usa el campo denominado Reservado para Uso del Sistema (BP 113-176), los 64 bytes que lo forman pueden almacenar información específica de un sistema. La propuesta HSG no establece el contenido de estos campos; sin embargo, no se deben almacenar datos para una aplicación determinada, ya que estos tienen un lugar reservado como se verá posteriormente.

V. Registro de Directorio. Un registro completo de directorio para el extent del archivo asociado con este XAR es duplicado en el XAR, con el objeto de asegurar que el XAR sea un lugar donde el sistema puede encontrar toda la información acerca del archivo. Algunos sistemas liberados pueden usar este registro de directorio extra para facilitar el acceso al archivo. El número de directorios padre también es proporcionado

para que el sistema tenga la habilidad de localizar un directorio padre dentro de la tabla de trayectoria.

VI. Información de Aplicación. El último campo en el XAR es de información para una aplicación especifica. La propuesta HSG no determina el tamaño de este campo, pero no puede ser mayor de 64 kbytes. El campo de Longitud Reservada para Uso de Aplicación (BP 247-250), indica que tamaño tendrá el área que usará la aplicación. y es la única especificación de la propuesta HSG que se necesita para poder usar esta área, sin embargo,la aplicación determina su contenido.

3.6 VOLUMEN DEL DISCO (The Volume Disc).

El área de un disco donde la información puede ser grabada se conoce como Espacio del Volumen (Volume Space), el cual está formado por todos los sectores disponibles para los que el Libro Amarillo Philips/Sony no define su uso.

El Espacio del Volumen se inicia en el sector lógico 0 y se extiende hasta el final del área grabada en el disco. Se divide en dos áreas; una del sistema y otra de datos; la primera, comprende del LSN 0 al LSN 15. La propuesta HSG no define su contenido, por lo cual, los diseñadores del disco pueden usarla como lo jusguen conveniente, pero asegurándose de que el sistema liberado tenga la inteligencia necesaria para interpretar y actuar sobre lo que se escriba en él. El área de datos ocupa todos los sectores restantes grabados en el disco.

3.6.1 Descriptor del Volumen (Volume Descriptor).

Un disco CD-ROM es conocido como un Volumen. Al inicio del área de datos de cada volumen hay una estructura que contiene el Descriptor del Volumen, el cual describe todo el contenido del disco, proporcionando información acerca de la organización lógica del disco, la localización del directorio raiz para la estructura estándar del sistema de archivos, el número y localización de algunas particiones no especificadas en el disco, los autores, la fecha y hora de creación, y mucho más. La secuencia del Descriptor del Volumen es la única parte del área de datos para la cual la propuesta HSG especifica su localización. El sistema puede encontrar la localización de otras estructuras importantes en el disco leyendo el Descriptor del Volumen.

3.6.2 Estructura del Descriptor de Volumen.

El Descriptor del Volumen consta de 2048 bytes, en registros de longitud fija, los cuales deben ser grabados secuencialmente,

iniciando el LSN 16. La secuencia del Descriptor del Volumen puede ser tan larga como sea necesario para describir todas las estructuras en el disco. El Terminador de la Secuencia del Descriptor del Volumen, indica el fin de la secuencia del Descriptor del Volumen.

Cinco tipos de Descriptores de Volumen están definidos en la propuesta HSG, los cuales son sintetizados en la fig. 3.6.2.1. La secuencia del Descriptor de Volumen puede contener varios tipos de Descriptores de Volumen en muchas combinaciones. Sin embargo, dos limitantes tienen lugar en la secuencia de organización:

I. Cada disco debe contener por lo menos un Descriptor de Volumen Estandar . Un disco con formato HSG debe tener la Estructura Estándar de Archivo y necesita un Descriptor de Volumen Estandar para localizar está estructura del archivo. Si el autor del disco pone más de un Descriptor de Volumen Estandar para obtener una redundancia en caso de un error de disco, cada uno debe hacer referencia a la misma Estructura Estándar de Archivo.

II. Al menos un Terminador de Secuencia del Descriptor del Volumen debe ponerse después de todos los Descriptores

Descriptor Estandar del Volumen	Secuencia terminadora del Volumen

Ejemplo 1

Registro 1	Descriptor Estandar del Volumen	Registro 2	Descriptor Estandar del Volumen	Secuencia terminadora del Volumen

Ejemplo 2

Conjunto de Caracteres #1	Registro	Descriptor Estandar del Volume	Descriptor del volumen no especif.	Conjunto de Caracteres #2	Secuencia terminadora del Volumen

Ejemplo 3

Figura 3.6.2.1 Ejemplos de secuencias de Descriptores de Volumen.

3.6.3 Estructura Estándar del Archivo Descriptor de Volumen (Standard File Structure Volume Descriptor).

La propuesta HSG requiere que cada disco contenga una Estructura Estandar de Archivo, ya que esto simplifica la jerarquía del directorio del sistema y la Tabla de Trayectoria que fueron descritas anteriormente. La Estructura Estandar del Archivo Descriptor de Volumen proporciona información esencial acerca de esta parte fundamental del formato del disco.

Para asegurarse que "los discos son intercambiables", los campos en esta estructura, contienen identificadores con arreglos de caracteres, que usan el conjunto de caracteres estándar ISO 646.

Los campos de la Estructura Estandar del Archivo Descriptor de Volumen muestran la flexibilidad de la propuesta HSG.

I. Identificador del Sistema (System Identifier). Este campo proporciona un medio para identificar el sistema que usa el área del sistema del disco. Si una implementación reconoce un identificador en esta zona, puede proceder a procesar los datos en el Area del Sistema.

II. Identificador de la Aplicación (Aplication Identifier) Una aplicación puede usar estos 128 bytes para decir como se interpretarán los datos en el disco.

III. Identificador del Conjunto del Volumen (Volume Set Identifier). Regresaremos a este campo cuando veamos los conjuntos de multivolúmenes; sin embargo, su primer uso es para almacenar el nombre de un conjunto de discos y además, es usado en conjunción con el Número de Secuencia del Volumen en el Conjunto (Volume Set Sequence Number), para construir conjuntos de varios discos.

IV. Registro de Directorio para el Directorio Raíz (Directory Record for Root Directory). Debemos conocer la localización del directorio raíz antes de poder trabajar con la estructura del directorio. Este registro de 34 bytes apunta hacia la raíz y es una forma de localizar el directorio raíz; una segunda forma se obtiene leyendo la Tabla de Trayectoria.

V. Identificador del Archivo de Derechos de Copia (Copyright File Identifier). Los autores de discos pueden crear un archivo para almacenar información de derechos de copia y poner el nombre del archivo en este campo. Este archivo se encuentra en el directorio raiz, y su nombre se limita a ocho caracteres y su extensión a

tres. Nombres reservados no son usados para prevenir conflictos con los que se asocian con otras aplicaciones o sistemas.

VI. **Identificador del Archivo Abstracto (Abstrac File Identifier)**. Este campo es usado y restringido en la misma forma que el anterior y ambos son tipo texto para facilitar su lectura y despliegue. El archivo abstracto puede contener la información que el autor quiera.

VII. **Conjunto de Caracteres Codificados en el Descriptor del Volumen (Coded Character Set Volume Descriptor)**. La propuesta HSG como ya vimos antes, limita los nombres de archivos y directorios a un subconjunto del ISO 646 para asegurar que los discos sean intercambiables; sin embargo, también reconoce que otros conjuntos de caracteres con frecuencia son deseables para nombres de archivos y directorios. Por tal razón, la propuesta permite a los autores de los discos, construir estructuras de directorio usando un Conjunto de Caracteres Codificados en lugar del Conjunto de Caracteres Estándar ISO 646.

Esta estructura alterna de directorio está físicamente separada de la Estructura Estándar de Archivo y utiliza diferentes archivos de directorio y directorio raiz. Sin embargo, su estructura tiene la misma forma que la Tabla de Trayectoria, los registros de directorio, etc.. Un sistema que puede leer correctamente la Estructura Estándar de Archivo, podrá leer también las estructuras de directorio asociadas con los conjuntos de caracteres de código, con solo pequeñas modificaciones de software.

El Conjunto de Caracteres Codificados en el Descriptor del Volumen proporciona información esencial acerca de cada conjunto de caracteres codificados del sistema de archivo. Este descriptor contiene dos importantes adiciones al Archivo Estandar del Descriptor del Volumen del Sistema.

1. El campo de Conjunto de Caracteres Codificados para el Identificador del Descriptor (Coded Character Set for Descriptor Identifier), especifica las secuencias usadas para interpretar los campos en el descriptor del Volume y los registros de directorio para el conjunto de caracteres codificados del sistema de archivo.
2. El Byte Bandera del Volumen (Volume Flag Byte: BP 16) indica si el campo del Conjunto de Caracteres Codificados tiene una o más secuencias de escape que no están registradas de acuerdo al ISO 2375.

VIII. Estructura no Especificada en el Descriptor del Volumen (Unspecified Structure Volume Descriptor). Este campo describe una área en el disco en la cual el autor del disco puede grabar datos que no forman parte de la Estructura Estándar de Archivo del HSG, y como los contenidos en dicha área no están especificados el autor puede usarla como mejor le parezca. Esta área es parte del Espacio del Volumen descrito anteriormente, y se pueden referenciar datos dentro de ella desde el Sistema Estándar de Archivos.

IX. Registro de Boot (Boot Record). El tipo final del Descriptor del Volmen de la propuesta HSG descrita es el Registro de Boot. Sistemas o aplicaciones específicas usan este descriptor para dar a su sistema de cómputo el estado deseado. Por ejemplo, un Registro Boot puede cargar un sistema operativo o un programa de aplicación asociado con el disco montado. Un disco puede tener un número ilimitado de Registros de Boot, pero la combinación de los campos: Identificador del Sistema de Boot (Boot System Identifier: BP 16-47) y el Identificador de Boot (Boot Identifier: BP 48-79) puede ser única para garantizar que el sistema anfitrión ó la aplicación puedan identificar apropiadamente el Boot Record.

3.7 CONJUNTOS MULTIVOLUMENES.

Aunque la capacidad del CD-ROM es muy grande, conjuntos de datos pueden facilmente llenar más de un disco. Como consecuencia, la propuesta HSG contempla combinar discos en conjuntos multivolumen.

La Estructura Estandar del Archivo Descriptor del Volumen, permite que el autor de disco especifique el nombre del conjunto y el número de volumenes en él. En general, dos situaciones propician la creación de un conjunto multivolumen. En la primera, el autor del disco tiene un conjunto de datos de un tamaño conocido que excede la capacidad de un disco. El número de discos requeridos por el conjunto de datos se conoce antes de que el conjunto de datos se encuentre en la última etapa en el proceso de construcción de un disco, en la que se obtiene el disco "MASTER" para obtener después todas las copias que se quieran; en la segunda, la situación se crea cuando los editores manejan conjuntos de datos que crecen con el tiempo, por lo cual, se deben adicionar discos periodicamente a un conjunto de Volumenes que ya existe.

3.7.1 Creación del Conjunto Multivolumen al "Mismo Tiempo".

Para el primer caso el editor debe definir el contenido de tres campos en el Descriptor Estandar del Volumen con los valores apropiados. Estos tres campos son los siguientes:

I. **Identificador del Conjunto al que pertenece el Volumen (Volume Set Identifier:** BP 215-342). Este campo contiene el nombre del conjunto al que pertenece el volumen, y todos los discos en el conjunto deben tener el mismo nombre, ya que así es como el sistema sabe si todos los discos montados pertenecen a un conjunto.

II. **Tamaño del Conjunto al que pertenece el Volumen (Volume Set Size:** BP 129-132). Este campo almacena un número que especifica cuantos discos contiene un conjunto, por lo cual, el valor mínimo que puede contener es 2, y el máximo es limitado por la propuesta HSG a 65535.

III. **Número Secuencial del Volumen en el Conjunto (Volume Set Secuence Number:** BP 133-136). Cada disco en el conjunto tiene un número diferente en este campo, iniciando la numeración con 1 y se continúa secuencialmente hasta llegar al valor especificado en el campo Tamaño del Conjunto al que pertenece el Volumen (Volume Set Size).

La propuesta establece para esta situación que cada disco en el conjunto contenga información sobre todos los discos del conjunto, por lo cual, cada disco deberá contener una Tabla de Trayectoria y una Estructura de Directorio en el mismo lugar y que sean válidos para todo el conjunto, lo que permitirá encontrar un archivo sin tener que montar todos los discos, sin embargo, no descarta la posibilidad de que para leer un archivo se tenga que usar más de un disco.

3.7.2 Actualización de Conjuntos Multivolumen.

El segundo caso es un poco más complejo, pero permite a un editor adicionar volumenes a un conjunto después de que los discos iniciales fueron hechos. Supongamos que tenemos un conjunto de dos vólumenes y posteriormente adicionamos otro volumen. Los valores para el Identificador del Conjunto del Volumen, Tamaño del Volumen (Volume Size), y otros campos importantes del Archivo Estandar de la Estructura del Descriptor del Volumen se muestran a continuación:

Conjunto No.	Volumen ID	Conjunto de IDs del Volumen	Definición del Tamaño del Conjunto	Conjunto de Secuencias del Volumen
Disco 1, conjunto inicial	dskorig_1	disco_Set	2	1
Disco 2, conjunto inicial	dskorig_2	disco_Set	2	2
Disco 3, adicionado posterior-mente	disc_adic	disco_Set	3	3

El Identificador del Volumen (Volume Identifier), como se observa varía de disco a disco, sin embargo, el Identificador del Conjunto al que pertenece el Volumen debe ser el mismo.

Para el conjunto inicial, la Definición del Tamaño del Conjunto al que pertenece el Vólumen , es de 2, y cada volumen en el conjunto original tiene un Número de Secuencia que Define al Volumen (Volume Set Sequence Number: VSSN) que indica su posición en el conjunto. Cuando se adiciona el nuevo volumen, el valor en su campo de Definición del Tamaño del Conjunto al que pertenece el Volumen, refleja esto, y su VSSN indica que él es el último en el conjunto. Notemos también que no podemos cambiar este campo de los volúmenes anteriores, por lo cual, debemos hacer que el nuevo volumen sea el único que refleje el estado actual del conjunto. La propuesta HSG especifica que el disco con el valor más alto en el campo del Conjunto del Tamaño del Volumen contenga la versión de la Tabla de Trayectoria y la estructura de directorio válida para el conjunto, haciendo que el directorio en los discos anteriores sea obsoleta. Este nuevo directorio es el único que puede proporcionar información acerca de todos los archivos en el conjunto.

Esta característica es muy poderosa ya que, nos permite "borrar" o "cambiar" archivos en los discos anteriores, con solo omitirlos en el directorio del último disco. Debemos tener presente que lo anterior no se hace físicamente, sino de una manera artificial y que si nosotros montamos primero un disco diferente al último, los archivos que se habían eliminado pueden habilitarse otra vez.

3.8 OPCIONES PARA EL ALMACENAMIENTO Y MANEJO DE ARCHIVOS.

La propuesta HSG proporciona muchas otras características opcionales que los autores pueden implementar, sin embargo, éstas no se pueden implementar en todos los niveles de la propuesta HSG. A continuación mencionaremos algunas de las más interesantes.

3.8.1 Archivos Intercalados.

Muchas aplicaciones en tiempo real requieren una velocidad especifica para la transferencia de datos desde el disco. La velocidad leyendo todos los sectores secuencialmente está fijada a 150 KB/seg, la cual es muy alta y puede sobrecargar la aplicación a menos que se pare la lectura. Un estudio cuidadoso de la tecnología de CD-ROM revela que cada vez que se detiene la lectura, se pierde el tiempo equivalente a casi dos revoluciones del disco antes de la siguiente lectura, lo cual es mucho tiempo para algunas aplicaciones. Por lo tanto, una aplicación en tiempo real no puede recibir los datos rápidamente o detener el flujo de datos y esperar, ya que, ninguna de las dos cosas es atractiva.

La propuesta HSG supera esta dificultad grabando los archivos como una secuencia continua de sectores, intercalando sectores vacios a intervalos cuidadosamente calculados entre los sectores grabados. De esta forma, el driver puede continuar leyendo el archivo secuencialmente, y con software de alto nivel o con hardware desechar los sectores vacios que no se quieren.

La Fig.3.8.1.1 muestra como un archivo es expandido en esta forma para reducir la velocidad de transferencia de datos.

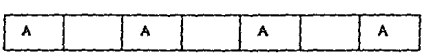


Fig. 3.8.1.1 Archivo con bloques vacios de desperdicio intercalados

Los cuadros marcados con la letra A representan a los sectores del disco que contienen datos del archivo A, y los cuadros vacios representan a los sectores que se les intercalaron.

La Fig. 3.8.1.2 ilustra un uso diferente del intercalado, donde se alternan los sectores de dos archivos, lo cual puede ser útil en algunas aplicaciones. Aqui leemos 3 sectores del archivo B y después leemos 2 sectores del C y luego regresamos al B, logrando con esto que el flujo de datos sea secuencial todo el tiempo.

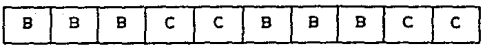


Fig. 3.8.1.2 Archivos con sus bloques de datos intercalados.

Dos campos en el registro de directorio especifican el intercalado de archivos. El primero de estos campos está etiquetado como Tamaño de Intercalado (Interleave Size: BP 27) y el segundo como Factor de Salto de Intercalado (Interleave Skip Factor: BP 29-32). El primer campo define el número de bloques lógicos consecutivos grabados en un archivo, y el segundo le indica al sistema cuantos bloques lógicos tiene que saltar antes de leer la siguiente parte del archivo en el inicio de un sector.

3.8.2 Archivos Asociados.

En algunas aplicaciones, tener dos o más archivos con el mismo identificador de archivo puede ser ventajoso, ya que, la relación que existe entre ellos es más estrecha que la que pueden tener dos archivos que tengan el mismo nombre y diferente extensión. Por ejemplo, un archivo puede contener el texto de un documento y el otro puede contener información de fuentes y formato. En esta situación, el bit del Archivo Asociado (bit posición 2) del campo de Banderas del Archivo en el registro de directorio es puesto en 1. El software de aplicación ó el del sistema liberado debe interpretar correctamente dicha bandera y actuar de acuerdo ella.

3.8.3 Archivos Ocultos (Hidden Files).

El byte (BP 25) de Banderas del Archivo en el Registro de Directorio también contiene un bit (bit posición 0) que permite a un archivo ser Oculto (hidden), en otras palabras, cuando el usuario utilice ciertos comandos para desplegar los archivos en un directorio, el sistema liberado no reconocerá la existencia de estos archivos, sin embargo, esta característica de los archivos depende del sistema liberado que se esté usando.

3.9 NIVELES DE INTERCAMBIO.

La propuesta HSG proporciona tres niveles de implementación de software para desarrolladores de sistemas, los cuales se llamaron "Niveles de Intercambio de Datos", y permiten a los autores hacer discos que incorporen todas o solo algunas de estas características propuestas según sus necesidades.

3.9.1 Nivel Uno.

Es el nivel más bajo de intercambio, incorporá un número mínimo de características, eliminando la mayoría de las características más sofisticadas de la propuesta. Un sistema CD- ROM (reproductor CD-ROM, computadora, sistema operativo y software para recuperar) que puede leer datos formateados de acuerdo a la propuesta HSG estará habilitado para leer estas características del Nivel 1 ocasionando las siguientes restricciones:

I. Están prohibidos los conjuntos multivolumen.

II. No se permiten archivos ocultos

III. Los archivos asociados están prohibidos. Esto se debe a que los sistemas del Nivel 1 no estarán habilitados para encontrar dos archivos compartiendo el mismo nombre en el mismo subdirectorio.

IV. Los campos de protección en el XAR también pueden ser ignorados.

V. No se permiten archivos con multiples extents.

VI. No se permite intercalar archivos.

VII. Los nombres de archivos y directorios se limitan a 8 caracteres.

VIII. Las extensiones de los archivos se limitan a 3 caracteres.

IX. El número de versión del archivo no está permitido.

3.9.2 Nivel Dos.

Este nivel proporciona características adicionales que la mayoría de los diseñadores necesitarán, especialmente aquellos que están trabajando con el formato estándar CD-I. Difiere del Nivel 1 únicamente en que permite los archivos intercalados y los nombres de archivos y directorios pueden tener hasta 31 caracteres; todas las otras restricciones del Nivel 1 son aplicables.

Los sistemas liberados en nivel 2 también podrán leer correctamente discos en Nivel 1.

La propuesta HSG sugiere que se pueden hacer textos sobre discos CD-ROM compatibles con CD-I usando las especificaciones de formato en Nivel 1 y Nivel 2 en conjunción con los códigos de caracteres ISO 646. El VTOC y la estructura de directorio del CD-I y el CD-ROM son completamente compatibles. Sin embargo, el formato del CD-I es más simple porque está diseñado para un sistema operativo determinado, por lo cual, no podrá actuar sobre el contenido de los XARs.

3.9.3 Nivel Tres.

Todas las características y capacidades de la propuesta HSG se proporcionan en este nivel, sin embargo, implementar discos completamente en él requiere un sistema operativo que pueda manejar características como archivos en multiextens, conjuntos de multivolumen e intercalado, ya que, si un sistema solamente ignora estos atributos especiales, puede leer los datos erroneamente, haciendolos inusables.

Los sistemas liberados en Nivel 3 también podrán leer correctamente discos hechos en los niveles 1 y 2, sin embargo, lo inverso no es cierto.

3.10 CONCLUSION.

La propuesta HSG fue formulada para proporcionar a las compañías que desean desarrollar aplicaciones en CD-ROM un formato estándar, el cual, puede ser manejado por varios sistemas liberados. En estos momentos nos encontramos en el proceso de estandarización (este proceso puede durar unos 5 años). Los desarrolladores de CD-ROM tuvieron que unir sus esfuerzos para poder hacer frente a lo que propone el HSG.

Para la mayoria de los publicadores de CD-ROM, su problema no consiste en como detectar los formatos de los discos, sino como determinar que características de la propuesta necesita un una publicación en especial. Después de que determinen que caracteristicas necesitan, deben encontrar un sistema que las pueda manejar.

CAPITULO IV

ANALISIS Y DISEÑO DEL SISTEMA

4.1 INTRODUCCION.

En este capítulo abordaremos el análisis y el diseño del sistema lector de CD-ROM. El análisis consta de varias etapas, en las cuales se considera toda la información que se pudo recopilar durante la investigación. Una de las etapas del análisis es el estudio de factibilidad en el que se determina si el proyecto se puede llevar a cabo o no; después, si el proyecto es factible se efectúa el análisis de las alternativas, mediante el cual se escoge la mejor solución para el problema y por último se analizan los requerimientos necesarios para llevar a cabo el proyecto.

Por otro lado, durante el el proceso de diseño se definen las funciones del sistema, así como, los distintos procesos que el sistema deberá realizar, dando una descripción de cada proceso junto con un diagrama de bloques del mismo. Por último, se hace una integración de todos los procesos para tener el sistema completo.

4.2 ANALISIS DEL SISTEMA

4.2.1 Estudio de Factibilidad.

Para llevar a cabo el desarrollo del manejador de CD-ROM, encontramos que algunos de los principales factores que determinan su factibilidad son la posibilidad de contar con los elementos de hardaware necesarios o en su defecto tener la posibilidad de desarrollarlos; tener acceso a las herramientas de software indispensables para el desarrollo del proyecto y los recursos económicos que nos permitan obtener los dos requerimientos anteriormente descritos.

Respecto al hardware, debido a que los trabajos sobre el CD-ROM son relativamente nuevos, existen en la actualidad pocos sistemas capaces de manejar esta nueva tecnología para almacenar información. Recordemos que las Computadoras Personales (PCs) ya controlan los discos CD-ROM utilizando el nivel más bajo de la propuesta HSG. Por otro lado de la investigación realizada, encontramos que el reproductor de CD-ROM que se ofrece en el mercado, está equipado con una interfaz SCSI para conectarse a la computadora y como la computadora que va a ser utilizada, basada en el microporcesador 68000 , está equipada con dicha interfaz para el control de sus discos de sistema, el reproductor de CD-ROM puede instalarse directamente como un disco de subsistema, sin tener incompatibilidad en las señales de control y de datos, lo que hace que en el proyecto se ahorren gastos en hardware y solo se tengan que invertir horas-hombres en el desarrollo del software requerido.

El sistema de cómputo a ser usado, es un sistema multiusuario que actualmente no cuenta con un controlador de CD-ROM efectivo, es por ésto que surgió la necesidad de desarrollar el presente proyecto, el cual fué abierto por la empresa que se encarga de la distribución en México y Latinoamerica del sistema de computó ya descrito.

Respecto al software que se requiere para el desarrollo del controlador, se tienen diferentes alternativas y no existen problemas por la utilización del mismo, ya que la empresa antes mencionada también tiene acceso al software para el sistema basado en 68000. Entre los lenguajes que podemos usar se encuentran Basic, C, Ensamblador 68000 y otros, sin embargo, los tres mencionados, son los que ofrecen más ventajas para el tipo de desarrollo que pretendemos realizar.

El análisis del factor económico, no se realizará. Para nuestro caso la factibilidad que resulta de dicho análisis, ya fue considerado por la empresa antes mencionada, desde el momento en que fue abierto el proyecto. De lo anterior podemos decir que el factor económico no será ningún impedimento para llevar a cabo el desarrollo del proyecto, ya que será financiado por la empresa.

El proyecto en un principio se limitará a trabajar solo con las caracteristicas del nivel uno de la propuesta HSG y posteriormente se podrán ir aumentando sus capacidades para trabajar con los siguientes niveles.

Por todo lo dicho anteriormente, podemos concluir que la realización del Controlador para CD-ROM es completamente factible.

4.2.2 Análisis de Alternativas.

Actualmente solo contamos con dos alternativas para desarrollar el sistema lector de CD-ROM, la primera es utilizar las utilerias que el sistema AMOS ofrece para el manejo del reproductor de CD-ROM y la otra es desarrollar nuestro propio driver y en ambos casos desarrollar las subrutinas necesarias para la recuperación de información del CD-ROM.

a) Análisis de las utilerias de AMOS.

El sistema de cómputo bajo el cuál se llevará a cabo el desarrollo de este proyecto, tiene diversas variantes basadas principalmente en el tipo de procesador que ulilice, la arquitectura del sistema, la configuración, y la versión del sistema operativo que esté usando. De acuerdo a estas caracteristicas los sistemas de computo se encuentran agrupados en varios modelos, por ejemplo: AM1000, AM1000/E, AM1000/XP, AM1200, AM1500, AM1545, AM2000, etc..

Basados en la información obtenida encontramos que la interfaz que utiliza el reproductor del CD-ROM es compatible con los siguientes sistemas:

AM1200
AM1500
AM2000

Si se quiere conectar a un sistema AM1000 con el microprocesador 68000 se tiene que hacer algunas pequeñas modificaciones a la tarjeta del CPU.

Uno de los mayores problemas con el CD-ROM es el formato estándar que se utiliza para escribir los datos.

La mayoria de los datos habilitados en el CD-ROM se encuentran formateados como un disco que va a ser manejado por el sistema operativo MS-DOS, ya que como recordaremos, en la actualidad la mayoria de los reproductores para CD-ROM se conectan a las PCs. Con cada CD-ROM se proporcionan las extensiones para MS-DOS para que éste maneje un tamaño más grande de bloques ya que normalmente maneja bloques de 512 bytes. AMOS también trabaja con bloques de 512 bytes, sin embargo, la forma en que funciona el Sistema de Servicio a Disco (DSKSER: Disk Service Sistem) implica un número muy grande de modificaciones en él para lograr que trabaje con bloques de 2048 bytes. Por lo tanto la decisión que se tomó está orientada en el sentido de forzar al reproductor de CD-ROM para que maneje bloques de 512 bytes.

Además el interactuar con dispositivos de acceso aleatorio que no tienen la estructura que maneja AMOS a través de lenguajes de alto nivel es difícil, por lo cual se ve la necesidad de desarrollar subrutinas en ensamblador para recuperar un bloque.

Una de las estrategias a utilizar es considerar que el reproductor de CD-ROM se comporta como un disco Winchester lo cual facilitará al usuario interactuar con él, utilizando las llamadas para acceso a los archivos que se proporcionan en todos los lenguajes de programación disponibles en AMOS con excepción de SMC-BASIC y UNIFY.

Una de las cosas que hace AMOS es simular que el CD-ROM posée los siguientes bloques:

```
Etiqueta del disco ( bloque 0 )
MFD ( Master File Directory ( bloque 1 ))
UFD ( User File Directory ( bloque 2 ))
```

Lo anterior se logra metiendo en el driver CDR.DVR una seudo copia de estos bloques, logrando con esto que el sistema crea que se trata de otro disco winchester y al tratar de leer dichos bloques el driver se los regresa al sistema.

Este metodo permite usar todos los comandos y subrutinas de AMOS para accesar la información de los discos en forma física, sin embargo, claramente se observa que no se tiene acceso real a los bloques 0, 1 y 2 del CD-ROM, los cuales pertenecen al sector lógico 0 y que según la propuesta High Sierra es parte del área de sistema, la cual no la estamos usando actualmente, dado que la información que nosotros necesitamos se encuentra en el área de datos que inicia en el sector lógico 16 que corresponde al bloque lógico 64, por tal razón no se observa ningún problema en usar las utilerías de AMOS.

En nuestro caso es importante el tener acceso a la verdadera etiqueta del disco CDROM, dado que ésta nos servirá para identificar que disco es el que estamos usando con solo leerla.

Las utilerías que nos ofrece AMOS permiten crear dos tipos de manejadores de discos (DRIVERS). Uno con el formato tradicional, que solo permite discos de hasta 32 MB, por este motivo para trabajar el disco CDROM se debe hacer mediante el uso de 19 unidades lógicas. El otro tipo de driver se puede hacer con formato extendido, el cual permite manejar al disco CDROM como una solo unidad lógica de 600 MB (todo el disco físico) o si se desea se puede dividir también en varias unidades lógicas.

Para crear el driver del CDROM, AMOS cuenta con el programa FIXCD.LIT. El cual al ejecutarse nos pide la siguiente información:

- Tipo de sistema de archivo en cual se quiere emular. De acuerdo al menu que presenta el "1" corresponde al sistema tradicional y el "2" al extendido.

- El número de subsitema (un valor entre 1 y 3 debido a que AMOS soporta 4 discos conectados al mismo controlador y el disco de sistema es el subsistema 0).

- Nombre del driver (es el nombre mediante el cual AMOS reconoce cual es el disco que debe accesar).

Por otro lado, antes de poder hacer uso del CDROM, es necesario definirlo en el sistema que va a manejarlo además de adicionar en la memoria del sistema el driver que generamos con FIXCD. Para que el sistema reconozca estos cambios, es necesario reinicializarlo.

b) Creación de un Driver nuevo.

La creación de un CDR.DVR nuevo solo se tomará en cuenta despúes de que se hagan todas las pruebas necesarias al que se tiene actualmente. Las necesidades que debe satisfacer el CDR.DVR son las siguientes:

- Direccionar adecuadamente los sectores del disco, lo cual se debe realizar en forma física.

- Poder manejar al CDROM como un conjunto finito de discos más pequeños, para aquellos sistemas que tengan versiones de AMOS que no manejen el formato extendido para direccionamiento de información en el disco. Aunque para esto se tienen que desarrollar comandos propios para cada uno de los dos formatos de disco.

Si después de realizar las pruebas no se satisface por lo menos la primera necesidad, se tendrá que plantear el desarrollo de otro CDR.DVR que si la satisfaga. Sin embargo, si alguna de las otras necesidades no se satisface, entonces podemos establecer ciertas condiciones para el hacer uso del CDR.DVR que ya está hecho.

Para poder crear un driver para el CDROM (CDR.DVR) es necesario conocer fisicamente los elementos que intervienen en la transferencia de información entre la computadora y el CDROM. Esto implica:

- Conocer las especificaciones físicas y tecnicas del disco.

- Conocer las señales que manejan para comunicarse el controlador del disco y la computadora (Interface SCSI).

- Conocer que puerto utiliza la computadora para comunicarse con los discos.

La mayoria de esta información ya se encuentra incluida en esta misma tesis, en la sección en la que se describio el disco, la computadora y la interface SCSI.

Por otro lado, también se debe conocer a fondo la forma en que AMOS maneja a los discos, el tipo de interrupciones que usa, la forma en que el driver interactua con el sistema cuando se realiza algúna operación que involucre a los discos, etc.. Aquí nos encontramos con pequeño escollo ya que parte de la información que necesitamos no es de dominio público, por lo que obtenerla se llevaría algún tiempo, el cual no se puede cuantificar.

Adicionalmente se necesita conocer muy bien las herramientas de software con las que AMOS cuenta para la realización de un driver para CDROM.

Como ya se menciono anteriormente, esta alternativa solo se tomará en consideración en el caso de que la primera no satisfaga nuestras necesidades.

4.2.3 Análisis del Sistema Lector de CD-ROM.

Los requerimientos necesarios para llevar a cabo el desarrollo de este proyecto, ya fueron mencionados en forma general en el análisis de factibilidad, sin embargo, después del análisis de alternativas que realizamos, podemos hacer un análisis de requerimientos donde se definen con detalle las caracteristicas que en nuestro caso nos interesan.

a) Requerimientos de Hardware.

A continuación se listarán los elementos de hardware que se utilizarán:

+ CPU con micorporcesador de la familia 68000

+ Fuente de Poder para satisfacer las necesidades de corriente de todos los circuitos y elementos que componen al sistema de cómputo.

+ Interfase SCSI

+ Periféricos, entre los que se incluyen terminales e impresoaras.

+ Reproductor de CD-ROM, el cual cuenta con una ranura para la introducción del CD en forma automática, una ranura para conectar el cable de corriente. Dos conectores hembras de 50 pines, uno para conectarse directamente a la interfaz del sistema y el otro para interconectarse con otro reproductor.

+ Cables para toma de corriente de 3 hilos para mayor protección de los equipos. Podemos aprovechar esta característica, dado que el lugar donde se desarrollará el proyecto cuenta con una instalación apropiada para el manejo de equipo de cómputo.

+ Cables planos de 50 pines con conectores tipo Centronics.

+ CD-ROMs disponibles para pruebas. En este punto cabe hacer notar que los CD-ROMs deben estar grabados de acuerdo al formato High Sierra.

b) Requerimientos de Software.

Como ya se mencionó anteriormente el sistema de cómputo en el cual se va a desarrollar el proyecto, es capaz de soportar varios lenguajes de programación y multiples sistemas de aplicación general, como son los procesadores de textos, editor general para los diferentes lenguajes, etc.

A continuación mencionaremos los sistemas de software que vamos a utilizar y las características principales por las que se eligieron:

+ Lenguaje ensamblador. Por el tipo de aplicación que vamos a desarrollar, el lenguaje ensamblador es el que más ventajas ofrece en cuanto a la obtención del máximo rendimiento que se puede tener del equipo, ya que aprovecha en forma más directa todas las capacidades que tiene el sistema para el manejo de sus recursos como son: pantalla, memoria, y los discos, que son los que más nos interesan en este momento. Además, este lenguaje nos permite utilizar las funciones más elementales del sistema mediante el uso de las llamadas a monitor, algunas de las cuales nos facilitan la realización de algunos procesos para tener acceso al disco. En resumen, el lenguaje ensamblador es la mejor opción para aquellas aplicaciones que tiene que ver con el hardware o con el sistema operativo del equipo de computo.

Por otro lado, una de las desventajas que tiene el utilizar un lenguaje de bajo nivel como el ensamblador, es que resulta un poco complicado la programación con él, a pesar de que se utilizen mnemónicos, además de que se requiere un mayor conocimiento del ensamblador, en comparación con los lenguajes de alto nivel, cuya cualidad principal es precisamente la de facilitar la programación.

No obstante, el sistema AMOS cuenta con otras herramientas que facilitan el trabajo cuando se utiliza ensamblador, por ejemplo, el AlphaFix que nos permite seguir paso a paso la ejecución de un programa, con la opción de interactuar con el sistema durante dicha ejecución para poder modificar las condiciones de éste, simulando así las condicones en las que debe operar el programa realmente. La presentación del programa a la hora de utilizar AlphaFix, es mediante el uso de mnemónicos, la utilización de direcciones de memoria absolutas y de programa para la localización de las instrucciones y datos del programa, pero con la opción de poder adicionar etiquetas para facilitar el seguimiento de la ejecución del programa en cuestión. También se ofrece un ligador (linker) de ensamblador, el cual nos permite unir programas pequeños para poder realizar funciones más complicadas, éste ayuda principalmente durante los procesos de prueba y detección de errores: por otro lado, se ofrece un generador de librerias que tiene que ver mucho con el ligador, ya que con el primero se pueden crear programas generales que se pueden utilizar en muchos otros.

Por los argumentos anteriores, podemos decir que el lenguaje con mayores probabilidades de utilizarse en el desarrollo del lector de CD-ROM es el ensamblador.

+ Lenguaje Basic. Este lenguaje nos ofrece una gran diversidad de aplicaciones, por lo cual podríamos pensar que la realización del proyecto se puede llevar a cabo en este lenguaje. La ventaja que nos brinda la utilización de un lenguaje de alto nivel como Basic, es que la sencilles para programar con él es muy grande, lo cual redundaria en un gran ahorro de tiempo en desarrollo y pruebas para detección de errores. Una de las principales desventajas que mencionamos, es el hecho de que los lenguajes de alto nivel tienen limitaciones para ciertas aplicaciones debido a que sus instrucciones están muy bien definidas y el programador no puede modificarlas. No obstante, el Basic que soporta Alpha es muy versatil y entre otras características nos ofrece la capacidad de poder llamar subrutinas en lenguaje ensamblador, lo cual es muy bueno, ya que como se menciono, los programas en ensamblador son más pequeños y rápidos que su equivalente en Basic, además de que algunos procesos son imposibles de realizar en un lenguaje de alto nivel.

En adición a las ventajas que se mencionaron ya del Basic, debemos agregar que este lenguaje se incluye en todos los equipos de computo Alpha como parte de su software básico, de esta manera si se cuenta con este último se cuenta también con Basic.

+ Lenguaje C. Este lenguaje posee características que nos brindan muchas facilidades para el desarrollo de nuestro proyecto, debido a que C hace una conjunción de las características que tienen los lenguajes de alto nivel y los de bajo nivel, es decir, C nos ofrece las ventajas de los dos lenguajes que vimos anteriormente. Sin embargo, aunque el sistema que vamos a utilizar también puede soportar el lenguaje C, el software para éste, como no es parte del sistema operativo, no es actualizado tan constantemente como el del sistema. Por otro lado, C es uno de de los tantos lenguajes que el sistema soporta, mientras que Basic es el lenguaje hecho especialmente para el manejo de la computadora que estamos utilizando.

Por todo lo que hemos dicho anteriormente podemos concluir lo siguiente:

El lector de disco CD-ROM, será realizado en lenguaje Basic y rutinas en ensamblador, con lo que obtendremos un manejador rápido y eficiente por la utilización del ensamblador, y sencillo de entender por el Basic.

Otros Requerimientos.

Basicamente nos referimos a los requerimientos que facilitan el desarrollo del proyecto pero que no son indispensables. Ejemplos de éstos son los siguientes:

+ Procesador de textos para la documentación del proyecto.

+ Papel de impresión para la documentación del proyecto por computadora.

+ Videocassettes para respaldar la documentación y programas que se vayan realizando.

+ Videocassettera para realizar los respaldos.

+ Manuales del software y hardware que se va a utilizar.

Como ya se mencionó anteriormente, todos los requerimientos son factibles de obtenerse.

4.3 DISEÑO DEL SISTEMA.

Basados en la información recopilada y sintetizada en los primeros capitulos de esta tesis, y en las conclusiones que se pueden obtener del análisis del sistema, es aquí donde se define, limita y estructura el sistema lector de CD-ROM.

Todo lo visto anteriormente es ahora utilizado en el diseño del sistema.

En esta etapa del desarrollo, el sistema se divide en diferentes niveles y procesos de una forma lógica, para obtener de esta manera una estructura que nos permita un desarrollo ordenado gradual y controlado.

El diseño se hace en tres fases de desarrollo: primero, enunciando los diferentes procesos de los que constará el sistema; segundo, describiendo en forma escrita y precisa cada proceso y tercero, describiendo al sistema de una manera gráfica, para su mejor comprensión.

4.3.1 Definición de las Funciones del Sistema

La definición de funciones es la determinación y limitación de lo que se quiere obtener del sistema.

Al definir las funciones de dicho sistema concretamos las acciones a realizar.

Para nuestro caso, las funciones que debe cumplir el sistema lector de CD-ROM son las siguientes:

I. Habilitar la comunicación entre la computadora y el reproductor de CD-ROM.

   Esto se refiere básicamente a lograr la comunicación física entre la computadora y el CD-ROM mediante el uso de la interfase SCSI.

   Para la comunicación física, es necesario conocer las características eléctricas y mecánicas tanto de la computadora como del reproductor del CD-ROM, además de que se debe contar con los elementos físicos para llevar a cabo la conexión, tales como cable, conectores, etc.

II. Habilitar el acceso adecuado a la información contenida en el CD-ROM.

Esto se refiere básicamente a la utilización de las utilerias del sistema para la generación del manejador del disco CD-ROM como un subsistema.

La creación del driver, como ya se ha definido en el análisis del sistema, se hará por medio del programa FIXCDR.LIT.

III. Generar rutinas y comandos que permitan el acceso a la información del disco.

A. Rutina para accesar los bloques o sectores físicos del disco. Esta rutina debe poder ser invocada desde un programa en lenguaje de alto nivel, ya que cualquier acceso lógico, que son a los que generalmente se refieren los programadores, siempre involucra un acceso físico.

B. Obtención del directorio del CD-ROM. Este es un comando muy útil para accesar la información que contiene el CD-ROM. También nos sirve para identificar al CD-ROM en caso de que por alguna razón no se identifiquen los discos por medio de una etiqueta.

C. Rutina para abertura de archivos. En este caso se trata de adaptar las instrucciones utilizadas para abrir archivos almacenados en los discos magnéticos, para que se puedan abrir archivos alamacenados en CD-ROM desde un lenguaje de alto nivel como BASIC.

D. Rutina para cerrado de archivos. Siempre que se tiene una instrucción de apertura debe existir otra de cerrado, la cual se encarga de restaurar el estado del sistema (registros y apuntadores) e inclusive para la liberación de memoria (por ejemplo, se libera la memoria que utiliza el sistema para manejar el archivo abierto).

E. Generación de otros comandos y rutinas generales y que puedan servir a los desarrolladores de software. Como son:

a) Comando para copiar archivos del CD-ROM a disco magnético, para una manipulación más rápida.

b) Comando para desplegar en pantalla archivos del CD-ROM.

c) Comando para conocer la etiqueta del disco.

Estos últimos comandos sí pueden ayudar en un momento dado, aunque en algunas ocasiones no se podrán aplicar debido a la naturaleza de la información de los CD-ROMs, en donde un solo archivo puede ser tan grande que ocupe más espacio del que pueda tener el disco magnético.

4.3.2. Definición de los Procesos del Sistema.

La definición de procesos es la división del sistema en actividades que guarden cierta relación entre sí.

De acuerdo a las funciones concretas definidas en el inciso anterior, podemos definir los procesos que formarán parte del sistema.

De esta manera tenemos los siguientes procesos:

Proceso de Conexión Física

Para poder cumplir con la primera función del sistema, es necesario sentar las bases sobre las cuales funcionará el sistema lector de CD-ROM.

El proceso, constará de los pasos a seguir para interconectar y comunicar los elementos para el sistema lector, es decir:

1. Verificar que el sistema de cómputo cumpla con las características para lograr la comunicación con el CD-ROM, las cuales son:

   a) El equipo debe contar con una interfase SCSI.

b) El equipo debe ser el adecuado, es decir, que sea de la familia 1200 o posterior, o en su defecto se debe verificar que la tarjeta de CPU tenga las modificaciones necesarias.

2. Verificar el reproductor del CD-ROM para determinar su compatibilidad con la computadora. Debe tener la interfase SCSI con un puerto para interconectarse con la computadora como un subsistema.

3. Verificar el tipo de conectores utilizados en ambas interfases SCSI para la elaboración del cable para la interconexión.

4. Verificar la instalación en general, que los voltajes, reguladores, cables, conectores, etc. sean los correctos.

5. Una vez hechas las verificaciones, se procede a instalar y conectar los elementos del sistema de acuerdo a las especificaciones indicadas en los manuales correspondientes.

A continuación, en la figura 4.3.2.1 presentamos un diagrama esquemático de los elementos y sus interrelaciones involucradas en este proceso.

En dicha figura podemos observar que aparte de los elementos ya mencionados tenemos otros periféricos como son la terminal y la impresora.

La terminal en este caso es indispensable, ya que a través de ésta damos las instrucciones a la computadora para que realize las acciones que se desean, y por otro lado en ésta se pueden visualizar los resultados.

La impresora es un elemento que nos ayuda a visualizar resultados. En el sistema lector se puede utilizar para imprimir información del disco: para el desarrollo del sistema nos facilita el trabajo al tener en papel los archivos de los programas, facilidad que se tendrá también durante el mantenimiento del sistema.

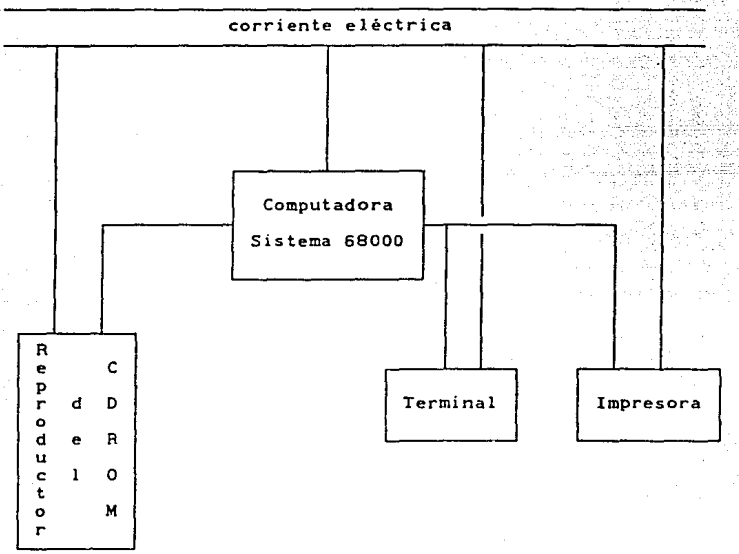


Figura 4.3.2.1

La configuración mostrada en el esquema anterior es la mínima necesaria para el desarrollo del sistema lector (aunque se podría prescindir de la impresora). Como el sistema de cómputo en el que nos basamos es multiusuario, puede haber más periféricos (terminales principalmente) conectadas a la computadora, lo cual facilita el desarrollo, ya que puede haber más de una persona trabajando al mismo tiempo con el CD-ROM.

Al mencionar que el sistema de cómputo es multiusuario, se debe preveer que todos los procesos que se van a programar tengan las siguientes características:

- Deben ser re-usables.

- Deben ser re-entrantes.

- Deben ser re-localizables.

El concepto de re-usable se refiere a que el programa pueda ser ejecutado más de una vez, sin tener que cargarlo nuevamente en memoria. Esto quiere decir que el programa no se modifica así mismo al ejecutarse.

El concepto de re-entrante significa a que varios usuarios pueden estar corriendo simultáneamente el mismo programa sin ningún problema.

El concepto de re-localizable se refiere a que todos los programas deben hacerse de tal manera que no esten sujetos a direcciones especificas de memoria, es decir, que la memoria se debe manejar en forma relativa, para que cualquier usuario pueda utilizarlos.

Proceso de Comunicación Lógica

Nos referimos a la comunicación lógica, como el proceso por medio del cual se logrará el enlace entre el sistema operativo y el CD-ROM.

La creación del driver o manejador del disco, es la principal actividad a realizar en este proceso.

De acuerdo al resultado del análisis de alternativas, el driver del disco va a ser generado por medio de las utilerías que para tal fin, ofrece el sistema operativo.

Los pasos a seguir en este caso son los siguientes:

1. Instalar la utilería y demás software que se requiera.

   Para este caso se tendrá que conectar un periférico, nos referimos al dispositivo utilizado como unidad de respaldo, en este caso una videocassettera.

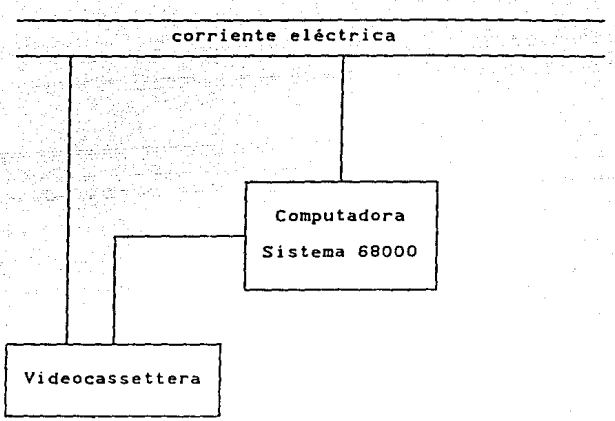


Figura 4.3.2.2

2. Invocar la instrucción adecuada que permita generar el driver. En este caso nos referimos a FIXCD.LIT.

   a) Teclear LOG DVR: ‹RET›

   b) Teclear FIXCD ‹RET›

   c) Definir las características con las que vamos a generar el driver del CD-ROM.

   d) Almacenar nombre para el driver.

   e) Salvar en el disco de sistema el driver de CD-ROM generado.

3. Definir a la computadora, los parámetros para que ésta identifique al reproductor de CD-ROM como dispositivo periférico del sistema.

   Para lograr lo anterior, hay que hacer algunas modificaciones a un archivo denominado AMOSL.INI al cual el sistema operativo AMOS identifica como archivo de inicialización y es en donde se definen las

características de la configuración del sistema que uno desea.

Las modificaciones a realizar son las siguientes:

a) Definir como dispositivos los discos lógicos que se hayan considerado para el CD-ROM:

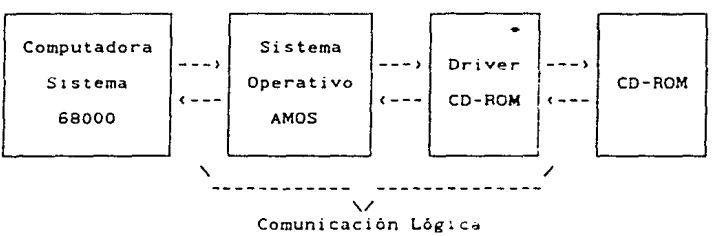
b) Cargar en la memoria del sistema, el driver del disco generado en el punto anterior.

4. Mediante los switches con los que cuenta el drive de CD-ROM, hay que asignarle el número de unidad que le corresponde en el bus SCSI, el cual debe concidir con el que se asigne en el punto (2).

5. Una vez que se tenga definido al CD-ROM, se le debe indicar al sistema que se va a trabajar con él. Esto se hace con un comando del sistema operativo denominado MOUNT y se le dá como parámetro el nombre de la unidad lógica con la que se va a trabajar.

6. Reinicializar el sistema(dar RESET a la computadora).

El objeto de este proceso es lograr la comunicación lógica, tal como se muestra en la figura 4.3.2.3.



* Proceso Generado.

Figura 4.3.2.3

Como se puede observar, en el diagrama anterior, son básicamente dos módulos los que conforman la comunicación lógica: el sistema operativo que ya existe y con el que se cuenta; y el driver que se genera.

Proceso para Accesar Físicamente al CD-ROM

Una vez que ya tenemos el driver del disco, el siguiente paso, es lograr el acceso a la información del mismo.

La información que se puede obtener de un disco puede ser de dos formas:

De forma física, cuando se hace referencia a bloques, tracks o sectores del disco.

De forma lógica, cuando se accesa por medio de archivos o registros, sin considerar en que parte del disco se encuentra la información.

El acceso lógico es la forma más común y sencilla para los usuarios, sin embargo todo acceso lógico involucra uno o más accesos físicos. Es por ésto que la base para leer el disco CD-ROM es lograr la lectura física.

El CD-ROM se puede leer en forma de bloques físicos de 512, 1024 y 2048 bytes.

Para nuestro caso, lo más conveniente es leer en bloques de 512 bytes, ya que es la unidad que el sistema operativo maneja. Sin embargo, se puede leer en bloques de 2048 bytes que es la forma en que la maneja el reproductor de CD-ROM.

De acuerdo a lo anterior, podemos tener diferentes procesos de acceso físico al CD-ROM:

I. Proceso para leer información del CD-ROM y dejar la almacenada en un archivo del disco magnético para su verificación posterior.

II. Proceso para leer información del CD-ROM y dejar la información en una variable que pueda ser manipulada por el programador.

Además de lo anterior, se pueden tener muchas variantes para accesar la información del disco, sin embargo las anteriores son representativas, con ellas se puede tener una manipulación más o menos completa de los datos.

Los pasos generales a seguir para la realización de éstos procesos, son los siguientes:

a) Pedir los datos de entrada, que en este caso serán: el número de bloque y la cantidad de éstos que se van a leer.

Esta etapa del proceso es muy fácil de realizar en un lenguaje de alto nivel como es Basic.

b) Definir el destino de la información, ya sea un archivo en el disco magnético o una variable.

c) Definir al CD-ROM como origen de la información.

d) Realizar las lecturas y escrituras correspondientes (lecturas al CD-ROM y escritura al archivo o a las variables).

Para el inciso (c) así como para el inciso (b) en caso de que se tenga que definir un archivo, debemos hacer uso del DDB (Data Device Block), el cual es utilizado por el sistema operativo para controlar el acceso a los archivos y dispositivos.

El DDB es una estructura creada en la memoria del usuario, en la que se definen los parámetros que identifican al dispositivo, se definen también los apuntadores a los bufferes de memoria para manipulación de la información, entre otras cosas. Para los archivos tenemos que proporcionar su especificación, su tamaño, el No. del bloque en el que inicia, etc.

El manejo del DDB debe hacerse con lenguaje ensamblador, dado que este lenguaje es el único que nos permite hacerlo en forma óptima.

Definidos el DDB y los parámetros de entrada, la lectura y escritura, se pueden hacer como si fueran operaciones para archivos convencionales.

De acuerdo a las características de las etapas, de los procesos de acceso físico, se realizarán con una mezcla de lenguaje de alto nivel, Basic, y lenguaje ensamblador.

La interacción con el usuario se realizará en Basic, debido a su sencilles y ràpidez para el desarrollo; mientras que las rutinas para creación y manipulación de archivos se hará en ensamblador.

Las rutinas en ensamblador se harán de tal modo que puedan ser llamadas desde el programa en Basic.

En la figura 4.3.2.4, se presenta un resumen en forma de bloques de las etapas para el proceso de acceso físico del CD-ROM.

Procesos para el Manejo de Datos del CD-ROM

Se puede decir que este proceso, corresponde a tener acceso lógico a los datos de CD-ROM.

Para este tipo de acceso al CD-ROM se pueden tener los siguientes procesos:

I. Acceso, obtención y manejo del Directorio.

Para este caso se requiere conocer la estructura y localización del directorio, lo cual ya fué definido en el capítulo 3 de esta tesis, donde se habla acerca del Formato Estándar High Sierra.

Por otro lado, en este proceso se hará uso de la rutina para acceso físico generada por el proceso anterior.

Este proceso puede tener dos opciones:

El de desplegar en pantalla el directorio.

El de crear un archivo en el disco magnético que contenga dicho directorio.

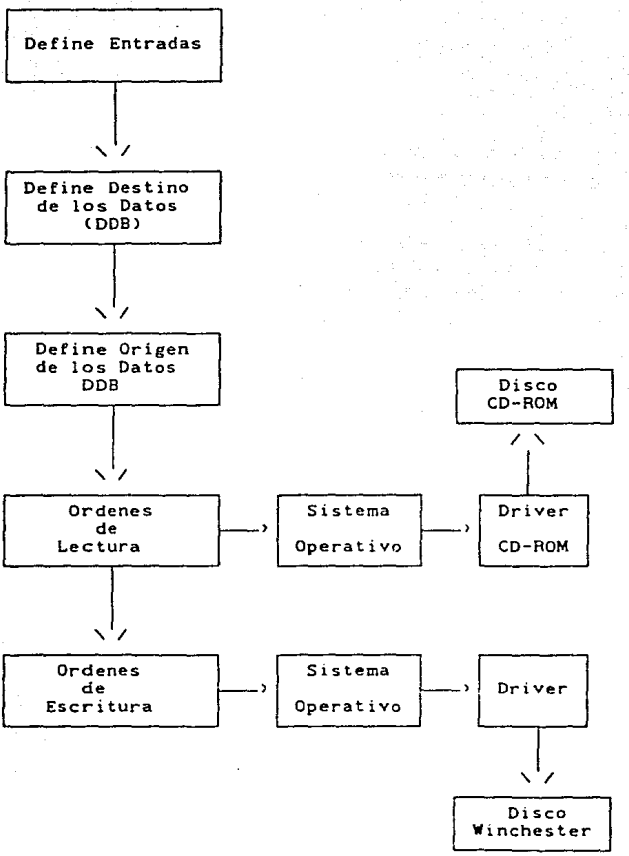

Define Entradas
Define Destino de los Datos (DDB)
Define Origen de los Datos DDB
Ordenes de Lectura
Sistema Operativo
Driver CD-ROM
Disco CD-ROM
Ordenes de Escritura
Sistema Operativo
Driver
Disco Winchester

Figura 4.3.2.4

El lenguaje más conveniente para realizar este proceso, resulta ser el ensamblador, ya que además de que se requiere manipular el DDB, este proceso puede quedar como un comando, los cuales son más eficientes si se hacen directamente con lenguajes como el ensamblador.

En la figura 4.3.2.5, se muestran las etapas del proceso de obtención del directorio.

Los pasos que se muestran son:

a) Definir al CD-ROM como origen, es decir, crear su DDB.

b) Definir como parámetros de entrada el bloque donde comienza el directorio. Esta información se puede obtener de la descripción del Formato High Sierra.

c) Utilizar la rutina para leer bloques físicos.

d) Controlar el despliegue o la escritura de la información.

II. Proceso de Abertura de Archivos.

El objeto de abrir un archivo, es el de determinar donde comienza y accesar la información deseada.

Por otro lado, al abrir un archivo se definen varios parámetros, entre los principales, se asigna un canal para acceso lógico, asociado a la descripción del archivo y al mismo tiempo el sistema crea un DDB en donde almacena todos los datos que necesita para accesar al archivo, como son: inicio, tamaño, tipo, número de bytes en el último bloque, etc.; también se define el área de memoria donde va a trabajar dicho archivo. Después el usuario hace referencia él mediante el número de canal que se le asigno.

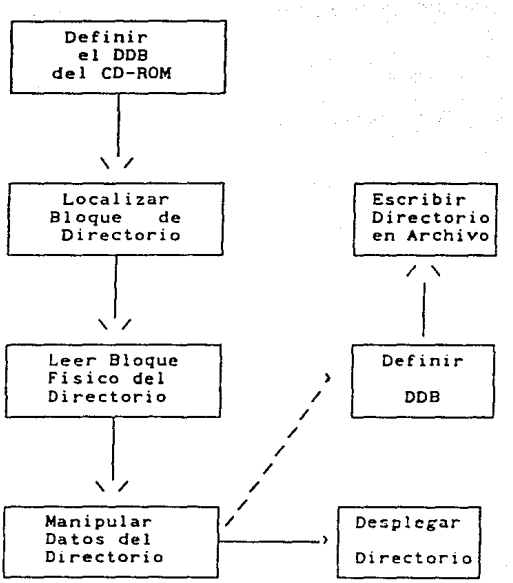

Definir
el DDB
del CD-ROM
Localizar
Bloque de
Directorio
Escribir
Directorio
en Archivo
Leer Bloque
Fisico del
Directorio
Definir
DDB
Manipular
Datos del
Directorio
Desplegar
Directorio

Figura 4.3.2.5

Los pasos que se requieren para la realización de este proceso, son los siguientes:

a) Definir al CD-ROM como dispositivo de lectura, es decir hay que crear su DDB y asignar un canal para acceso lógico.

b) Localizar a través del directorio, los datos del archivo que se desea accesar (inicio, tamaño, etc.).

En el siguiente diagrama de la figura 4.3.2.6 se muestran los pasos anteriores.

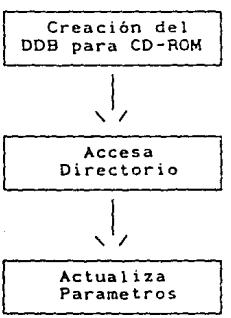


Figura 4.3.2.6

III. Proceso de Cerrado de Archivos.

Este proceso se requiere como oponente al anterior.

La función principal de este proceso, consistirá básicamente en eliminar el DDB generado por la apertura, y con ello liberar la memoria que tenía asignada el arhivo, así como su canal de comunicación.

IV. Otros Procesos.

Aquí solo se mencionará en general lo que deben considerar los procesos a realizar con objeto de que el sistema incluya las funciones definidas en el último punto.

Como ya se había mencionado, respecto a que el sistema considere funciones tales como el Copiar a disco magnético, el desplegar en pantalla un archivo y leer la etiqueta del disco. Estas funciones no son precisamente necesarias, ya que con los procesos ya considerados, de alguna manera se cubren los puntos que abarcarían estas otras funciones.

El proceso general que tiene que seguir estas rutinas es el que se muestra en la figura 4.3.2.7.

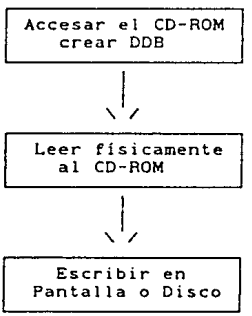


Figura 4.3.2.7

4.3.3 Estructura del Sistema

En el diagrama de la figura 4.3.3.1 se muestra la estructura general de sistema lector de CD-ROM basado en un sistema con microprosesador 68000.

Los dos primeros bloques del diagrama representan los elementos físicos principales del sistema. Todos los demás bloques representan procesos.

Como se puede observar, la mayoría de los procesos que se muestran en el diagrama, ya han sido descritos en las etapas anteriores del diseño. El único proceso que se visualiza y del que no se había hecho mención alguna es el último bloque, pero ésto es debido a que se refiere a aplicaciones específicas, y la finalidad de nuestro sistema es proporcionar las rutinas generales para que los programadores puedan hacer sus propios desarrollos.

El software correspondiente al último proceso puede involucrar la presentación de menús y la realización de rutinas muy particulares a la información que contenga el CD-ROM, por ejemplo, si el disco contiene referencias bibliográficas, el software nos podría proporcionar opciones para consultar por autor, por título, editorial, etc.

Como podemos observar este último proceso es muy variado por lo que ya no se considera dentro de los objetivos de nuestro proyecto, debido a que pretendemos dar herramientas generales que sirvan para cualquier tipo de aplicación y no solo para una en particular.

Con todo lo anterior hemos asentado lo que corresponde al diseño del sistema, y se ha visto el contexto en el que se encuentra éste.

El siguiente paso es llevar a la práctica todo lo visto hasta el momento.

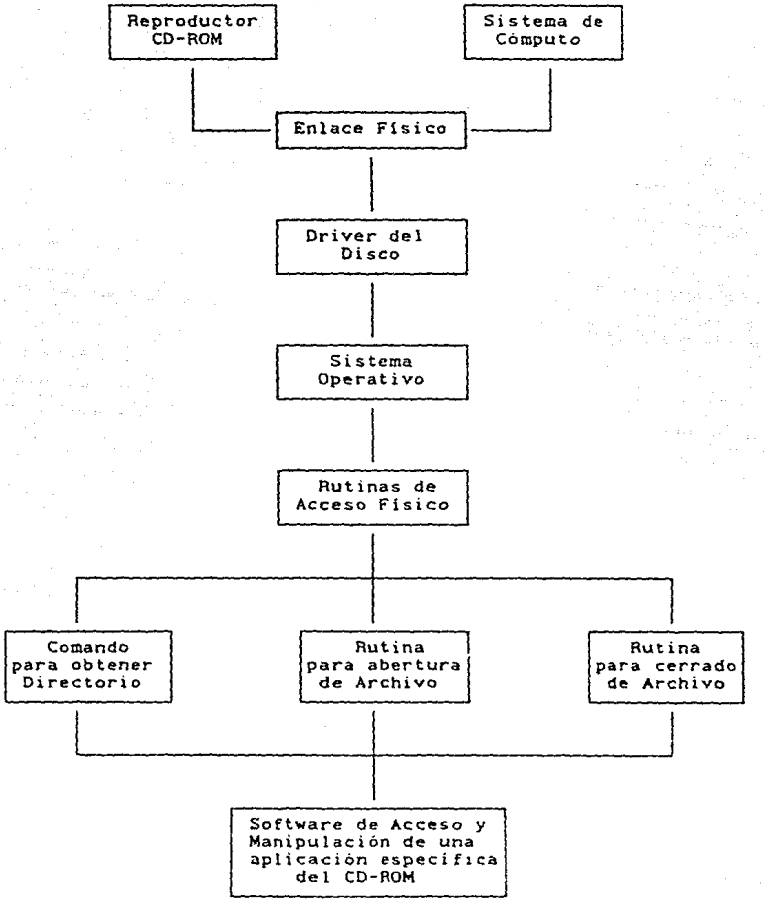

Reproductor CD-ROM
Sistema de Cómputo
Enlace Físico
Driver del Disco
Sistema Operativo
Rutinas de Acceso Físico
Comando para obtener Directorio
Rutina para abertura de Archivo
Rutina para cerrado de Archivo
Software de Acceso y Manipulación de una aplicación especifica del CD-ROM

Figura 4.3.3.1

CAPITULO V

DESARROLLO DEL SISTEMA Y RESULTADOS

5.1 INTRODUCCION.

En este capitulo de desarrollo y resultados, la mayor parte de las actividades que se describen son las correspondientes a las cuestiones prácticas.

Todo lo que se detalló en las etapas correspondientes al análisis y diseño del sistema se llevará a cabo.

En el texto de este capitulo se resumen los detalles que surgieron al realizar los procesos descritos en el diseño del sistema. También se enuncian las modificaciones que se tuvieran que hacer.

Este capitulo, además de ser una descripción de todas las actividades que se realizaran, tiene también por objeto servir de base a la elaboración de un documento de guía para los usuarios del CD-ROM con la computadora de sistema 68000 que ya hemos descrito, a través del sistema lector que se les proporcione.

Esto significa que aquí se encontrará la información necesaria para la instalación, uso y mantenimiento del sistema lector de CD-ROM, tanto de software como de hardware.

Este capítulo consta básicamente de cuantro incisos. En el primero se proporcionan los pasos y actividades a seguir para que el sistema de cómputo reconozca al CD-ROM como un dispositivo más que puede manejar; en el segundo, se presentan los listados de los programas hechos de acuerdo a lo especificado en el diseño; en el tercero, se describen todas las pruebas que se realizan a cada uno de los procesos que conforman al sistema, para verificar su correcto funcionamiento, y finalmente se presentan los resultados obtenidos.

5.2 INSTALACION DEL CD-ROM AL SISTEMA

El sistema se refiere a la computadora con microprocesador 68000 que se va a utilizar.

Esta computadora puede manejar una gran variedad de dispositivos periféricos, tales como terminales, impresoras, unidades de respaldo entre las que se encuentran los discos flexibles, cintas, etc. De éstos, el de mayor importancia es el disco duro, dado que es en éste donde se almacena el sistema operativo, el cual es indispensable para el funcionamiento de la computadora.

La computadora tiene la capacidad de manejar cuatro discos con el mismo controlador, cada uno de los cuales se identifican como subsistemas 0,1,2 y 3. El disco que contenga el sistema operativo debe ser siempre el subsistema 0, el cual también se conoce como disco de sistema, dado que la arquitectura usada en la computadora siempre hace referencia al disco de sistema para obtener la información necesaria para su funcionamiento.

Por otro lado, los discos de subsistema solo tienen la restricción de que al elaborar su driver, los datos proporcionados al sistema como son su capacidad y el número de subsistema que se le asigne en el controlador, coincidan con los que se tienen en el disco que se desea manejar. Cabe hacer notar que puede existir un subsistema 3 sin que existan necesariamente el subsistema 1 o 2.

En nuestro caso particular, el disco CD-ROM es considerado por la computadora como el subsistema 2.

La instalación de cualquier dispositivo, ya sea el disco del sistema, algún subsistema o cualquier otro periférico, involucra en la mayoría de las ocasiones los siguientes tres pasos:

1. Instalación Física.

2. Generación del Driver.

3. Definición del dispositivo en el archivo de inicialización del sistema.

El CD-ROM no es la excepción, su instalación completa consiste de lo siguiente:

1. Instalación Física o del Hardware.

   a) Se hace el cable de acuerdo a las señales del Bus SCSI.

      El cable que se utiliza, es un cable plano que conecta las señales en forma directa. Esto se puede utilizar por que la computadora y el drive del CD-ROM utilizan la misma interfaz.

      Un diagrama de las señales de la interfaz SCSI, lo tenemos en el Capitulo I página 26 de esta tesis.

      Los conectores a utilizar son de tipo Centronix. Los pines a conectar se pueden obtener de la tabla que para este efecto se muestra en el Capitulo I página 28 de esta tesis.

   b) Se define el número de subsistema que corresponde al CD-ROM por medio de los switches que se encuentran en su drive.
      Para definir dicho número, algunos discos utilizan "jumpers". En nuestro caso, este número se define por medio de switches.

La figura 5.2.1 trata de representar la parte frontal del drive del CD-ROM. Los switches se encuentran en esta cara del drive en el ángulo superior izquierdo.

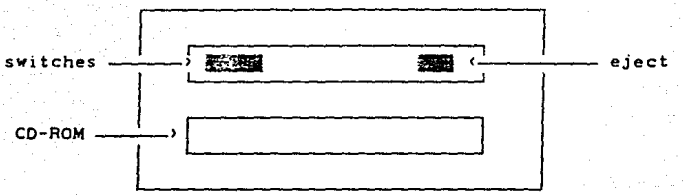


Figura 5.2.1

Para nuestro caso particular, la configuración de los switches se muestra en la figura 5.2.2.

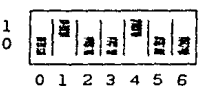


Figura 5.2.2. Esta figura nos muestra la codificación de un 2 binario por medio de los tres primeros switches (0,1 y 2) para indicar que el CD-ROM es el subsistema 2. Los otros switches son de control: el 3 es de Paridad; el 4 es de Arbitrio; el 5 y 6 son de Prueba.

c) Se preparan las cabezas del drive del CD-ROM por medio del tornillo que para este fin se tiene en la parte inferior del drive.

d) Se apagan el drive del CD-ROM y la computadora (en caso de que estuvieran encendidos).

e) Se interconectan el drive del CD-ROM a la computadora por medio del cable hecho en el inciso (a).

f) Se conectan el drive y la computadora a la corriente eléctrica.

g) Habilitar a la computadora con al menos una terminal para poder continuar con la instalación.

2. Generación del Driver del CD-ROM.

Para poder generar el driver, es necesario primero tener las utilerías en el disco del sistema (DSK0).

Dichas utilerías se encuentran almacenadas en videocassette.

El disco del sistema está dividido en áreas de trabajo. Las áreas de trabajo que nos interesan en este momento, son la del Operador (DSK0:[1,2]) identificada como "OPR:"; y el área de Drivers (DSK0:[1,6]) identificada como "DVR:".

Los pasos a seguir para la generación del driver, son los siguientes:

a) Teclear LOG OPR: ‹RET› para trabajar en el área del operador que es privilegiada.

b) Teclear VCRRES DSK0:[]=ALL:[] ‹RET› para copiar de videocassette a disco (restaurar) las utilerías.

c) Teclear LOG DVR: ‹RET› para trbajar en el área de drivers, que es donde el sistema busca los manejadores de los diferentes dispositivos con los que cuenta el sistema.

d) Teclear FIXCD ‹RET› que es el nombre del programa que crea el driver.

A continuación mostramos la información que despliega el programa y como deben ser llenadas las solicitudes de éste para nuestro caso:

FIXCD.LIT Version 1.0(100)

1. Traditional (AMOS 1.3) file system.

2. Extended (AMOS 2.0) file system.

Which file system would you like CD-ROM to emulate ? 2

Enter subsystem number (0-3): 2

Enter new driver name: CDR

New driver now in memory

El número de subsistema que pide, es el mismo número que se define con los switches del driver.

e) Teclear SAVE CDR.DVR ‹RET› por que el driver que genera FIXCD lo deja en memoria RAM, y con SAVE lo grabamos en el disco.

3. Definir el CD-ROM como dispositivo del sistema.

Para que el sistema reconozca al CD-ROM como dispositivo del sistema, es necesario que encuentre ciertas definiciones al respecto de éste, en el archivo de inicialización.

Al archivo de inicialización, que ya se tenga, se le deben agregar las siguientes líneas:

a) DEVTBL CDR0 para indicar a la computadora que el CD-ROM es un dispositivo del sistema, anexándolo a la Tabla de Dispositivos.

b) SYSTEM DVR:CDR para que el driver del CD-ROM se cargue a la memoria del sistema.

Una vez hecho todo lo anterior, hay que reiniciar el sistema.

5.3 PROGRAMACION DE LOS PROCESOS DEL SISTEMA

La programación de los procesos del sistema ya definidos, se refiere a la elaboración del archivo que contenga el programa fuente, del cual posteriormente se obtiene el programa ejecutable para su puesta en operación.

La creación del programa fuente, sin importar el lenguaje en el que se realice, requiere de la utilización de un Editor. El sistema operativo AMOS, cuenta con un editor denominado VUE, el cual es un editor de pantalla.

La instrucción para crear un archivo es:

VUE nombre.extensión <RET>

El nombre se refiere a un conjunto de caracteres alfanuméricos con una longitud variable entre 1 y 6 caracteres; y la extensión es de 1 a 3.

La extensión que se utiliza, dependen del lenguaje del programa: M68 si es un programa en ensamblador o BAS si es un programa en Basic.

Después de invocar al editor VUE, podemos teclear el programa de acuerdo al formato requerido para cada lenguaje.

Una vez que se tiene los fuentes de los programas, el siguiente paso a realizar, depende del lenguaje.

Si se tiene un programa en Basic, lo más conveniente es compilarlo. Decimos que es lo más conveniente, por que Basic es un intérprete, sin embargo este lenguaje en la computadora que usamos, es muy poderoso y entre otras ventajas, nos ofrece la oportunidad de compilar los programas.

Una característica importante del Basic de esta computadora, es que nos permite definir variables (MAP) de diferentes tipos y a diferentes niveles (variables que agrupan a otras). Además permite el uso de etiquetas evitando la necesidad de la numeración de lineas.

Para compilar un programa fuente en Basic, se teclea lo siguiente:

COMPIL nombre

La instrucción anterior, genera un archivo con el mismo nombre que el del programa fuente, pero con extensión .RUN, el cual se puede ejecutar con la instrucción:

RUN nombre

Si se trata de un programa en ensamblador, éste debe er ensamblado con la instrucción:

M68 nombre

El ensamblador de la computadora es muy poderoso, es simbólico, tiene definidas macroinstrucciones y llamadas a monitor o interupciones, que nos facilita ciertas operaciones con el sistema operativo y con los dispositivos de la computadora.

Como resultado del ensamblado, se obtiene un programa con el mismo nombre que el programa fuente pero con extensión .LIT. Sin embargo, el ensamblador permite que el usuario cambie el nombre o la extensión según sus necesidades. Por ejemplo, en el caso de que el usuario elabore una subrutina, ésta deberá tener la extensín .SBR.

Los programas con extensión .LIT, AMOS los interpreta como comandos o programas independientes, los cuales para ejecutarse solo se necesita especificar el nombre del programa seguido de sus parámetros separados por comas.

Por otro lado, en el caso de los programas con extensión .SBR, AMOS los reconoce como subrutinas, las cuales solamente se ejecutarán cuando sean llamadas desde algún programa.

Los programas ensamblados y con extensión .SBR, deben ser invocados en nuestro caso desde un programa Basic de la siguiente manera:

XCALL nombre,parámetros

Los procesos que se harán como comandos son los siguientes:

- Obtención del Directorio.
- Desplieque de Etiqueta.
- Copiado a Disco Magnético.
- Desplieque de Archivos en Pantalla.

Los procesos que se harán como subrutinas son los siguientes:

+ Acceso Físico.
+ Apertura de Archivos.
+ Cerrado de Archivos.

Aparte de lo anterior, un programa en ensamblador puede ser depurado o rastreado con el debbuger denominado FIX.

```
FIX nombre[.lit]
        o
FIX nombre.sbr
```

La extensión de default es .LIT.

En las siguientes páginas presentamos los listados de los programas realizados.

```
**********************************
**                              **
**   COMANDO PARA OBTENER  LA   **
**   ETIQUETA DEL DISCO CDROM   **
**                              **
**  Nombre: LABCD               **
**  Autor:  Jose Luis Damian G. **
**          Ricardo Zaldivar G. **
**  Fecha:  30/agosto/89        **
**                              **
**********************************


; PROGRAMA PARA OBTENER LA ETIQUETA DEL CDROM Y DESPLEGARLA EN
; PANTALLA

    SEARCH     SYS
    SEARCH     SYSSYM
    SEARCH     TRM
    ASMMSG


    RADIX      16
;
.OFINI
.OFDEF    CDR.DDB,D.DDB              ; DDB CONTROL BLOCK
.OFSIZ    CDR.SIZ

.OFINI
.OFDEF    LABEL.,48.                 ; Offset de la etiqueta
.OFSIZ    LABEL
.OFINI
.OFDEF    LAB.SIZ,16.                ; Tama&o de la etiqueta
.OFSIZ    LABSIZ


START:PHDR -1,0,PH$REE!PH$REU        ; Marca el programa como
                                     ; reentrante y reusable
;
;
; Al se utiliza para indexar el DDB para el CDROM
;
;
    GETIMP     CDR.SIZ,A1,NOMEM      ; Localizo un DDB en memoria
;
;
    INIT       CDR.DDB(A1)           ; Inicializamos el DDB
```

```
INICIO:
;
    MOVW        #1,D0                   ;numero de blocks a leer
    MOVW        #0,D4                   ;numero de record random
    MOV         #64.,D3                 ;inicio de lectura
    LEA         A2,RENAME               ;Indexamos al CDR0:
    FSPEC       CDR.DDB(A1)


5$:
    MOV         D3,D.REC(A1)
    READ        CDR.DDB(A1)
    MOV         D.BUF(A1),A2

; Inicio el despliegue

    MOVW        #<-1_8.>+0,D1           ; borra pantalla
    TCRT
    MOVW        #0120,D1                ; fija posicion de cursor
                                        ; en 1,34
    TCRT
    MOV         #LABSIZ,D2              ; Fija tama&o de la etiqueta
    ADD         #LABEL,A2               ; Fija inicio de la etiqueta
10$:
    MOVB        (A2)+,D1
    TOUT                                ; Despliega byte en pantalla
    DBF   D2,10$                        ; Si no has terminado continua
    CRLF
SAL:
    EXIT
RENAME:   ASCIZ      /CDR0:/            ; Nombre para respaldo
          EVEN
;

NOMEM:    EXIT
;
END
```

```
                ****************************************
                ** COMANDO PARA OBTENER EL DIRECTORIO **
                ** DEL DISCO CDROM                    **
                **                                    **
                ** Nombre: DIRCD                      **
                ** Autor:  Jose Luis Damian G.        **
                **         Ricardo Zaldivar G.        **
                ** Fecha:  30/julio/89                **
                **                                    **
                ****************************************


; PROGRAMA PARA OBTENER EL DIRECTORIO DEL CDROM Y DESPLEGARLA  EN
; PANTALLA

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG


    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;
.OFINI
.OFDEF   DIR.DDB,D.DDB              ; DDB CONTROL BLOCK
.OFSIZ   DIR.SIZ
;
:Generamos offset para el record de directorio
;
.OFINI
.OFDEF   NAM.FIL,12.
.OFDEF   BLK.FIL,4
.OFDEF   BLK.INI,4
.OFSIZ   REC.SIZ
;
START:   PHDR -1,0,PH$REE!PH$REU
;
; A1 libre para usarse en cualquier cosa
; A2 indice temporal, y apunta a la direccion del  nombre  del
;      archivo en CDROM
; A3 libre para usarse en cualquier cosa
; A5 registro para indexar el DDB de DIRECT.CDR
; A6 registro de scratch
;
INICIO:
;
```

```
; Generamos el DDB para accesar el directorio

    GETIMP      DIR.SIZ,A5,NOMEM

; Inicializamos el DDBs

    INIT        DIR.DDB(A5)

; Cargamos los DDBs con las especificaciones de los archivos

1$:

; Desliegue de encabezado

    CRLF
    LEA         A2,HEADER
38$:
    CMPB        (A2),#0
    BEQ         39$
    MOVB        (A2)+,D1
    TOUT
    BR          38$

;Busqueda de los datos del archivo que se desea abrir en el CDROM
;desde BASIC

39$:
    CRLF
    LEA         A2,NAME                 ; Indexa archivo de directoric
    FSPEC       DIR.DDB(A5)             ; Pasa el nombre al DDB
    MOVW        REC.SIZ,D.RSZ(A5)       ; Fija el tama&o del
                                        ; registro a leer
    CLR         D2
    OPENR       DIR.DDB(A5),F.WAT       ; Abrimos DIRECT.CDR
12$:
    MOV         D2,D.REC(A5)            ; Fija el numero de record a
                                        ; leer
    INPUT       DIR.DDB(A5)             ; Leemos DIRECT.CDR
    MOV         D.BUF(A5),A2            ; Indexa buffer de DIRECT.CDR
                                        ; con A2
    MOV         #24.,D3                 ; Fija contador de registros
                                        ; de DIREC
;   MOV         #1,D0                   ; Fija contador de caracteres
                                        ; = 0
;   POP         A0                      ; Restaura direccion de la
                                        ; especificion del archivo del
                                        ; CDROM
;   PUSH        A0                      ; Guarda especificacion del
                                        ; archivo
                                        ; que se busca en : DIRECT.CDR
8$:
    MOV         A2,D4                   ; D4 se usa para actualizar A2
    CMPB        (A2),#0                 ; Es el fin del DIRECT.CDR
```

```
    JEQ       SAL

; Inicia despliegue de directorio

40$:
    CMPB      (A2),#'.          ; (A2) = .
    BEQ       41$               ; Si, salta
    MOVB      (A2)+,D1
    TOUT
    BR        40$
41$:
    MOV       #3,D0
42$:
    MOVB      (A2)+,D1
    TOUT
    CMPB      (A2),#0
    BEQ       43$
    DBF       D0,42$
43$:
    MOV       D4,A2             ; Regresa direccion de
                                ; paramentros del archivo
                                ; localizado en DIRECT.CDR
    MOV       BLK.FIL(A2),D0    ; Pasa tama&o del archivo a D0
    SWAP      D0
    CLR       D7
    MOVW      D0,D7             ;
    MOV       #9,D6
    LSR       D0,D6             ; Hacemos division entre 512
                                ; rotando
    ANDW      #1FF,D7           ; La division es exacta
    BEQ       11$               ; Si, salta
    ADD       #1,D0             ; No, suma uno al cosiente
11$:
    TAB
    TAB
    MOV       D0,D1
    DCVT      10,OT$TRM         ; Convierte a decimal D1
    MOV       BLK.INI(A2),D0
    SWAP      D0
;
    TAB
    MOV       D0,D1
    DCVT      10,OT$TRM
    CTRLC     SAL
    MOV       D4,A2
    ADD       REC.SIZ,A2        ; Apunta al siguiente registro
    DBF       D3,8$
    ADD       #25,D2
    JMP       12$

SAL:
    CRLF
    CLOSE     DIR.DDB(A5)
    EXIT
```

```
;
RENAME:  ASCIZ     /CDR0:/          ; Nombre para respaldo
    EVEN
NAME:    ASCIZ     /DIRECT.CDR/     ; Nombre del archivo de
                                    ; directorio
         EVEN
HEADER:  ASCIZ     /   NOMBRE      No. BLOCKS   BLOCK INICIAL/
         EVEN
;
NOMEM:   EXIT
;
END
```

```
;**********************************
;**                              **
;**  COMANDO PARA DESPLEGAR UN   **
;**     ARCHIVO EN PANTALLA      **
;**                              **
;**  Nombre: TYPECD              **
;**  Autor:  Jose Luis Damian G. **
;**          Ricardo Zaldivar G. **
;**  Fecha:  24/agosto/89        **
;**                              **
;**********************************


; PROGRAMA PARA DESPLEGAR INFORMACION DEL CDROM EN PANTALLA

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG


    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;

.OFINI
.OFDEF    CDR.DDB,D.DDB              ; DDB para el CDROM
.OFSIZ    CDR.SIZ
;
.OFINI
.OFDEF    DIR.DDB,D.DDB              ; DDB para el directorio
.OFSIZ    DIR.SIZ

;Generamos offset para el record de directorio
.OFINI
.OFDEF    NAM.FIL,12.
.OFDEF    BLK.FIL,4
.OFDEF    BLK.INI,4
.OFSIZ    REC.SIZ

START:    PHDR -1,0,PH$REE!PH$REU

; A1 libre para usarse en cualquier cosa
; A2 indice temporal, y apunta a la direccion del nombre del
;    archivo en CDROM
; A3 registro para indexar el DDB del archivo abierto en CDRO
; A5 registro para indexar el DDB de DIRECT.CDR
; A6 registro de scratch
```

```
INICIO:
; Generamos los DDBs para los dos archivos

    GETIMP     CDR.SIZ,A3,NOMEM
    GETIMP     DIR.SIZ,A5,NOMEM

; Inicializamos los dos DDBs

    INIT       CDR.DDB(A3)
    INIT       DIR.DDB(A5)

; Cargamos los DDBs con las especificaciones de los archivos

1$:
    BYP
    LIN
    BNE     3$
    TYPECR     (Falto especificar el archivo)
    EXIT
3$:
    BYP                              ; Pasa blancos
    USRFRE     A0                    ; Indexa inicio de la memoria
                                     ; libre a A0
    PUSH       A0                    ; Guarda en el stack el inicio
5$:
    LIN
    BEQ        4$
    MOVB       (A2)+,(A0)+
    BR    5$
4$:
; Se termino de cargar nombre del archivo

    LEA        A2,NAME               ; Indexa archivo de directorio
    FSPEC      DIR.DDB(A5)           ; Pasa el nombre al DDB
    MOVW       REC.SIZ,D.RSZ(A5)     ; Fija el tama&o del registro a
                                     ; leer
    CLR        D2
    OPENR      DIR.DDB(A5),F.WAT     ; Abrimos DIRECT.CDR
12$:
    MOV        D2,D.REC(A5)          ; Fija el numero de record  a
                                     ; leer
    INPUT      DIR.DDB(A5)           ; Leemos DIRECT.CDR
    MOV        D.BUF(A5),A2          ; Indexa buffer de DIRECT.CDR
                                     ; con A2
    MOV        A2,D1                 ; D1 se usa para actualizar A2
    MOV        #25.,D3               ; Fija contador de  registros
                                     ; de DIREC
    MOV        #1,D0                 ; Contador de caracteres = 0
    POP        A0                    ; Restaura direccion de la
                                     ; especificion del archivo  del
                                     ; CDROM
```

```
    PUSH        A0                  ; Guarda especificacion  del
                                    ; archivo que se busca  en
                                    ; DIRECT.CDR
8$:
    CMPB        (A2),#0             ; Es el fin del  DIRECT.CDR
    JEQ         6$                  ; Si,salta
    CMPB        (A2),#'.            ; Se encontro el punto
    JNE         16$                 ; No, continua verificando
                                    ; nombre

; se inicia verificacion de extension
    CMP         D0,#8.              ; Es el numero de  caracteres
                                    ; del nombre <=  que el
                                    ; permitido
    BGT         17$                 ; Si continua verificando Ext.
    TYPECR      <Nombre de archivo fuente ilegal>
    EXIT
17$:
    MOV         #2,D0               ; Fija contador de Char. para
                                    ; la EXT
    INC         A2
    INC         A0
18$:
    CMMB        (A2)+,(A0)+         ; Compara Extension
    BNE         22$                 ; Salta al siguiente Reg.
    DBF         D0,18$              ; Sigue verificando Ext.
    BR          21$                 ; Se encontro el registro

16$:
    CMMB        (A2)+,(A0)+         ; Compara Nombre
    BEQ         7$                  ; Salta si no es igual
22$:
    ADD         REC.SIZ,D1          ; Incrementa apuntador para el
                                    ; siguiente  registro  de
                                    ; directorio
    MOV         D1,A2               ; Pasa apuntador a A2
15$:
    POP         A0                  ; Restaura direccion de la
                                    ; especificion del archivo del
                                    ; CDROM
    PUSH        A0                  ; Guarda especificacion del
                                    ; archivo que se  busca en
                                    ; DIRECT.CDR
    DBF         D3,8$
    ADD         #25,D2
    JMP         12$
7$:
    INC         D0                  ; Incrementa contador de
                                    ; caracteres
    JMP         8$
21$:
    CLOSE       DIR.DDB(A5)
    POP                             ; Libera el Stack
    BR          9$
```

```
6$:
    TYPECR      <No se encontro el archivo en el directorio del
CDROM>
    EXIT
9$:                                 ; Continua
; fijamos parametros en el DDB del archivo del CDROM

    LEA         A2,RENAME
    FSPEC       CDR.DDB(A3)
    ANDB        #^C<D$BYP>,D.FLG(A3); Habilitamos mensajes de
                                    ; error
    MOV         D1,A2               ; Regresa  direccion  de
                                    ; paramentros del archivo
                                    ; localizado en DIRECT.CDR
    MOV         BLK.FIL(A2),D0      ; Pasa tama&o del archivo a D0
    SWAP        D0
    CLR         D7
    MOVW        D0,D7
    MOV         #9,D6
    LSR         D0,D6               ; Hacemos division entre 512
                                    ; rotando
    ANDW        #1FF,D7             ; La division es exacta
    BEQ         11$                 ; Si, salta
    ADD         #1,D0               ; No, suma 1 al cociente
11$:
    MOV         D0,D2               ; Pasa tama&o del archivo a D2
    DEC         D2                  ;
    MOV         D0,D.FSZ+CDR.DDB(A3); Fijamos el tama&o del
                                    ; archivo
    MOV         BLK.INI(A2),D0
    SWAP        D0
    MOV         D0,D.BAS+CDR.DDB(A3); Block inicial
; se inicia el proceso de despliegue en la pantalla
    CLR         D4                  ; Contador de bytes en el
                                    ; archivo
20$:
    MOV         D.BUF(A3),A2        ; Indexa buffer con A2
    MOV         D0,D.REC(A3)        ; Fija record a leer
    CTRLC       ABORTA              ; Termina ejecucion por : ^C
    READ        CDR.DDB(A3)         ; Lee el CDROM
    MOV         #511.,D3            ; Fija numero de caracteres que
                                    ; se enviaran a la pantalla
31$:
    MOVB        (A2)+,D1            ; Pasa byte a desplegar en D1
    TOUT                            ; Despliega byte en D1
    INC         D4                  ; Inc. contador de bytes
    CTRLC       ABORTA              ; Si hay ^C termina
    DBF         D3,31$              ; Si se vacio el buffer  sal
                                    ; del loop
    INC         D0                  ; Inc. numero de block
                                    ; a leer en el CDROM
    DBF         D2,20$              ; Si es el fin del archivo
                                    ; termina la ejecucion
    CRLF
```

```
    TYPESP    (Numero de bytes en el archivo = )
    MOV       D4,D1
    DCVT      0,OT$TRM              ; No. de bytes desplegados
    CRLF
    EXIT
ABORTA:
    CRLF
    TYPESP    (Numero de bytes leidos antes del ^C = )
    MOV       D4,D1
    DCVT      0,OT$TRM              ; No. de bytes desplegados
    CRLF
SAL:
    EXIT
;
RENAME:  ASCIZ     /CDR0:/          ; Para uso del CDROM
NAME:    ASCIZ     /DIRECT.CDR/     ; Nombre del archivo de
                                    ; directorio
         EVEN
;
NOMEM:   EXIT
;
END
```

```
;*************************************
;**                                 **
;**  COMANDO PARA COPIAR UN ARCHIVO **
;**     DEL CDROM AL DISCO DURO     **
;**                                 **
;**  Nombre: COPYCD                 **
;**  Autor:  Jose Luis Damian G.    **
;**          Ricardo Zaldivar G.    **
;**  Fecha:  23/agosto/89           **
;**                                 **
;*************************************


; PROGRAMA PARA COPIAR INFORMACION DEL CDROM A UN ARCHIVO

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG


    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;
.OFINI
.OFDEF    DSK.DDB,D.DDB             ; DDB CONTROL BLOCK
.OFSIZ    DSK.SIZ

.OFINI
.OFDEF    CDR.DDB,D.DDB             ; DDB CONTROL BLOCK
.OFSIZ    CDR.SIZ
;
.OFINI
.OFDEF    DIR.DDB,D.DDB             ; DDB CONTROL BLOCK
.OFSIZ    DIR.SIZ

;Generamos offset para el record de directorio
.OFINI
.OFDEF    NAM.FIL,12.
.OFDEF    BLK.FIL,4
.OFDEF    BLK.INI,4
.OFSIZ    REC.SIZ


START:    PHDR -1,0,PH$REE!PH$REU
```

```
; A1 registro para indexar el DDB del archivo abierto en DSKn
; A2 indice temporal, y apunta a la direccion del nombre del
; archivo en CDROM
; A3 registro para indexar el DDB del archivo abierto en CDRO
; A5 registro para indexar el DDB de DIRECT.CDR
; A6 registro de scratch


INICIO:

; Generamos los DDBs para los dos archivos

    GETIMP    DSK.SIZ,A1,NOMEM
    GETIMP    CDR.SIZ,A3,NOMEM
    GETIMP    DIR.SIZ,A5,NOMEM

; Inicializamos los dos DDBs

    INIT DSK.DDB(A1)
    INIT CDR.DDB(A3)
    INIT DIR.DDB(A5)

; Cargamos los DDBs con las especificaciones de los archivos

1$:
    BYP
    FSPEC     DSK.DDB(A1),CDR      ; Espera archivo con Ext.   por
                                   ; default CDR
    LOOKUP    DSK.DDB(A1)          ; Existe el archivo
    BNE       2$                   ; No, salta
    TYPECR  (El archivo destino ya existe)
    EXIT
2$:
    CMPB      (A2)+,#'=            ; Se encontro el igual
    BEQ       3$                   ; Si, continua
    TYPECR    (Falto el archivo fuente)
    EXIT
3$:
    BYP                            ; Pasa blancos

    USRFRE    A0                   ; Pasa inicio de la memoria
                                   ; libre a A0
    PUSH      A0                   ; Guarda en el stack el  inicio
5$:
    LIN
    BEQ       4$
    MOVB      (A2)+,(A0)+          ;
    BR        5$
4$:

; Se termino de almacenar la especificacion del archivo
; Busqueda de los datos del archivo que se desea abrir en el
; CDROM desde BASIC
```

```
; A0 apunta al nombre del arhivo del CDROM especificado en el
; comando
; A2 sirve para indexar la FSPEC, y para indexar el buffer de
; DIRECT.CDR
; D0 es registro temporal para pasar valores al DDB y como
; contador
; D1 se utiliza como respaldo de A2 cuando funciona como el
; indice del buffer
; D2 se utiliza para pasar el No. de registro a leer en el
; DIRECT.CDR
; D3 se utiliza como contador de registros en el buffer

    LEA       A2,NAME              ; Indexa archivo de directorio
    FSPEC     DIR.DDB(A5)          ; Pasa el nombre al DDB
    MOVW      #REC.SIZ.,D.RSZ(A5)  ; Fija el tama&o del registro a
                                   ; leer
    CLR       D2
    OPENR     DIR.DDB(A5),F.WAT    ; Abrimos DIRECT.CDR
12$:
    MOV       D2,D.REC(A5)         ; Fija el numero de record a
                                   ; leer
    INPUT     DIR.DDB(A5)          ; Leemos DIRECT.CDR
    MOV       D.BUF(A5),A2         ; Indexa buffer de DIRECT.CDR
                                   ; con A2
    MOV       A2,D1                ; D1 sera usado para actualizar
                                   ; A2
    MOV       #25.,D3              ; Fija contador de registros de
                                   ; DIREC
    MOV       #1,D0                ; Fija contador de caracteres =
                                   ; 0
    POP       A0                   ; Restaura direccion de la
                                   ; especificion del archivo del
                                   ; CDROM
    PUSH      A0                   ; Guarda especificacion del
                                   ; archivo buscado en DIRECT.CDR
8$:
    CMPB      (A2),#0              ; Es el fin del DIRECT.CDR
    JEQ       6$                   ; Si,salta
    CMPB      (A2),#'.             ; Se encontro el punto
    JNE       16$                  ; No, continua verificando
                                   ; nombre

; se inicia verificacion de extension

    CMP       D0,#8.               ; Es el numero de caracteres
                                   ; del nombre (= que el
                                   ; permitido ?
    BGT       17$                  ; Si, continua con Ext.
    TYPECR    <Nombre de archivo fuente ilegal>
    EXIT
17$:
    MOV       #2,D0                ; Fija contador de Char. para
                                   ; la EXT
    INC       A2
```

```
    INC         A0
18$:
    CMMB        (A2)+,(A0)+         ; Compara Extension
    BNE         22$                 ; Salta al siguiente Reg.
    DBF         D0,18$              ; Sigue verificando Ext.
    BR          21$                 ; Se encontro el registro

16$:
    CMMB        (A2)+,(A0)+         ; Compara Nombre
    BEQ         7$                  ; Salta si no es igual
22$:
    ADD         #REC.SIZ.,D1        ; Incrementa apuntador para el
                                    ; siguite registro de
                                    ; directorio
    MOV         D1,A2
15$:
    POP         A0                  ; Restaura direccion de la
                                    ; especificion del archivo del
                                    ; CDROM
    PUSH        A0                  ; Guarda especificacion del
                                    ; archivo que se busca en
                                    ; DIRECT.CDR
    DBF         D3,8$
    ADD         #25,D2
    JMP         12$
7$:
    INC         D0                  ; Inc. contador de Carac.
    JMP         8$
21$:
    CLOSE       DIR.DDB(A5)
    POP                             ; Libera el Stack y A0
    BR          9$
6$:
    TYPECR      <No se encontro el archivo en el directorio del
CDROM>
    EXIT
9$:
; Se encontro el archivo desea, y continua el proceso
; fijamos parametros en el DDB del archivo del CDROM
; y en el archivo en el que se va a hacer la copia

    LEA         A2,RENAME
    FSPEC       CDR.DDB(A3)
    ANDB        #^C<D$BYP>,D.FLG(A3); Habilitamos mensajes de
                                    ; error
    MOV         D1,A2               ; Regresa direccion de
                                    ; paramentros del archivo
                                    ; localizado en DIRECT.CDR

    MOV         BLK.FIL(A2),D0      ; Pasa tama&o del archivo a D0
    SWAP        D0                  ; Invierte words en D0
    CLR         D7
    MOVW        D0,D7
    MOV         #9,D6
```

```
    LSR         D0,D6                   ; Hacemos division entre 512
                                        ; rotando
    ANDW        #1FF,D7                 ; La division es exacta
    BEQ         11$                     ; Si, salta
    ADD         #1,D0                   ; No, suma uno al residuo
11$:
    MOV         D0,D.FSZ+CDR.DDB(A3); Fijamos el tama&o del
                                        ; archivo en el DDB
;;  MOV  D0,D.FSZ+DSK.DDB(A1)           ; Fijamos el tama&o del
                                        ; archivo
                                        ; que se creara en el Disco
;;  MOV  D0,D.ARG(A1)                   ; Fijamos tama&o del archivo a
                                        ; crear en el disco DSK

    MOV  #100,D.ARG(A1)                 ; TAMA&O DE ARCHIVO PROVICIONAL

    DSKCTG      DSK.DDB(A1)             ; Crea Archivo en el disco
    MOV         #-1,D.LSZ+CDR.DDB(A3)
                                        ; Indica que el archivo en el
                                        ; CDROM es Random.
    MOV         BLK.INI(A2),D0
    SWAP        D0
    MOV         D0,D.BAS+CDR.DDB(A3): Block inicial

; se inicia el proceso de copiado del CDROM al DISCO

    MOV         #255.,D2
    MOV         #0,D3
    MOV         D.BUF(A1),D.BUF(A3) ; Comparte buffer de los dos
                                    ; archivos que se van a
                                    ; utilizar en la copia
    OPENR       DSK.DDB(A1)         ; Abre el archivo destino

    MOVW        #0FF00,D1           ; Limpia pantalla
    TCRT
    MOVW        #0101,D1            ; Cursor en 1,0
    TCRT
    MOVW        #0FF0B,D1           ; Fija baja intensidad
    TCRT
    TYPESP      (Grabando el bloque # )
    MOVW        #0FF0C,D1           ; Quita baja intensidad
    TCRT
    MOVW        #0116,D1            ; Cursor en 1,22
    TCRT

20$:
    MOV         D3,D.REC(A1)        ; Fija record a grabar
    MOV         D0,D.REC(A3)        ; Fija record a leer

    MOVW        #0116,D1            ; Cursor en 1,22
    TCRT
    MOV         D3,D1
    DCVT        0,OT$TRM            ; Despliega No. de record
```

```
    CTRLC     ABORTA              ; Termina ejecucion por ^C
    READ      CDR.DDB(A3)         ; Lee el CDROM
    OUTPTL    DSK.DDB(A1)         ; Graba en el DSK
    INC       D3
    INC       D0
    DBF       D2,20$
SAL:
    CLOSE     DSK.DDB(A1)         ; Cierra archivo en DSK
    EXIT

ABORTA:
    DSKDEL    DSK.DDB(A1),F.WAT   ; Borra el archivo que se creo
    EXIT
;
RENAME:
    ASCIZ     /CDR0:/             ; Nombre para respaldo
NAME:
    ASCIZ     /DIRECT.CDR/        ; Nombre del archivo de
                                  ; directorio
    EVEN
;
NOMEM:
    EXIT
;
END
```

```
;*****************************************
;**                                     **
;**  SUBRRUTINA PARA HACER LECTURAS EN  **
;**  EL DISCO CDROM EN FORMA FISICA  Y  **
;**  COPIAR LA INFORMACION A UN ARCHIVO **
;**                                     **
;**  Nombre: XREADC                     **
;**  Autor:  Jose Luis Damian G.        **
;**          Ricardo Zaldivar G.        **
;**  Fecha:  7/agosto/90                **
;**                                     **
;*****************************************


; PROGRAMA PARA OBTENER INFORMACION DEL CDROM Y COPIARLA EN UN
; ARCHIVO ABIERTO EN BASIC

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG
    OBJNAM    .SBR                  ; Cambia la extension .LIT por
                                    ; .SBR
    EXTERN    $FLSET
    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;
.OFINI
.OFDEF   DDB.CDR,D.DDB              ; DDB CONTROL BLOCK
.OFSIZ   CDR.SIZ

.OFINI
.OFDEF   DDB.BAS,D.DDB

;Generamos el offset para las varibles

.OFINI
.OFDEF   ARG.XC,2                   ; Offset contador de
                                    ; argumentos
.OFDEF   ARG.TYP,2                  ; Offset de tipo
.OFDEF   ADD.XC,4                   ; Direccion de argumento
.OFDEF   SIZ.XC,4                   ; Tama&o del argumento

.OFINI
.OFDEF   NUM.BLK,4                  ; Numero de bloques
.OFDEF   BLK.INI,4                  ; Bloque inicio
.OFDEF   FIL.CHN,2                  ; Numero de canal
```

```
.OFSIZ   ARG.SIZ

START:   PHDR -1,0,PH$REE!PH$REU

; A0 contiene la direccion deL inicio del area impura
; A1 registro para indexar el DDB del archivo abierto en basic
; A2 indice temporal, y apunta al DDB del archivo abierto en
;    BASIC
; A3 contiene la direccion de los argumentos
; A4 apunta a la base de la memoria libre
; A5 apunta al final de la memoria libre, se ocupara para indexar
;    el DDB
; A6 registro de scratch


    MOV       A5,D4
    SUB       A4,D4
    CMP       D4,#CDR.SIZ
    BGT       1$
    TYPECR    ‹No hay memoria suficiente para correr el XCALL
XREAD›
    EXIT
1$:
    MOV       A4,A1                 ; Indexa area libre en A1
    ADD       #CDR.SIZ,A4           ; Quita de la mem. libre
                                    ; un #DDB

    TSTW      ARG.XC(A3)            ; Hay argumentos ?
    BNE       2$                    ; Si,  continua
    TYPECR    ‹Faltan Argumentos en el XCALL XREAD›
    EXIT
2$:
    CMPW      ARG.TYP(A3),#0        ; Tipo de argumento correcto ?
    BEQ       3$                    ; Si, continua
    TYPECR    ‹Tipo de argumento invalido en el XCALL XREAD›
    EXIT
3$:
    CMP       SIZ.XC(A3),#ARG.SIZ   ; Es correcto el tama&o del
                                    ; argumento ?
    BEQ       4$                    ; Si, continua
    TYPECR    ‹Tama&o del argumento invalido para el XCALL
XREAD›
    EXIT
4$:
    PUSH      A5                    ; Guardamos la direccion del
                                    ; fin de memoria
    MOV       ADD.XC(A3),A5         ; Indexa direccion de
                                    ; argumentos
    CLR       D1
    MOVW      FIL.CHN(A5),D1
    CALL      $FLSET
    BEQ       5$                    ; Canal de archivo correcto ?
```

```
    TYPECR     ‹Se proporciono al XCALL XREAD un Canal de Archivo
invalido ›
    EXIT

5$:
    PUSH       A2                    ; Guardamos direccion del DDB
                                     ; del BASIC, A2 indexa el DDB
                                     ; del  canal abierto

INICIO:
    LEA        A2,RENAME             ; Indexa el dispositivo CDRO:
    FSPEC      DDB.CDR(A1)

; Procesamos los argumentos mandados desde basic

    MOV        NUM.BLK(A5),D0        ; Numero de bloques a  leer
    SWAP       D0                    ; Invertir posicion de  words
                                     ; en D0
    DEC        D0
    MOV        #0,D4                 ; Numero de record random
    MOV        BLK.INI(A5),D3        ; Inicio de lectura
    SWAP       D3                    ; Invierte posicion de  words
    POP        A2                    ; Restaura indice del DDB  de
                                     ; BASIC
; Definicion de parametros  para el DDB del CDROM
    MOV        D.BUF(A2),D.BUF(A1)   ; Compartimos buffer con  los
                                     ; 2 DDB's
    MOV        D.SIZ(A2),D.SIZ(A1)   ; Fijamos tama&o del buffer
                                     ; para CD
    MOVB       #‹D$INI!D$ERC!D$BYP›,D.FLG(A1)
                                     ; Fijamos banderas
    MOVW       #0FF00,D1             ; Limpia pantalla
    TCRT
    ANDB       #^C‹D$BYP›,D.FLG(A2); Habilitamos mensajes de
                                     ; error de AMOS
; Se inicia la escritura
    CRLF
    MOVW       #140A,D1              ; Renglon,columna
    TCRT
    MOVW       #‹-1_8.›+11,D1        ; Fijar baja intensidad
    TCRT
    TYPESP     ‹Leyendo de bloque # ›
    MOVW       #160A,D1              ; Renglon,columna
    TCRT
    TYPESP     ‹Grabando de bloque # ›
    MOVW       #‹-1_8.›+12,D1        ; Fijar display normal
    TCRT
    MOVB       #4,D.OPN(A2)          ; Fija bandera como OPENR
5$:
    MOV        D3,D.REC(A1)          ; Numero de Record a leer
    MOV        D4,D.REC(A2)          ; Numero de Record a grabar
    MOVW       #141E,D1              ; Renglon,columna
    TCRT
    CLR        D1
```

```
        MOV     D3,D1
        DCVT    0,OT$TRM
        READ    DDB.CDR(A1)          ; Lectura del CDROM

        OUTPTL  DDB.BAS(A2)          ; Escritura en el archivo  de
                                     ; BASIC
        MOVW    #161F,D1             ; Renglon,columna
        TCRT
        CLR     D1
        MOV     D4,D1
        DCVT    0,OT$TRM
        INC     D3
        INC     D4
        DBF     D0,5$
        CRLF
        MOVB    #24,D.OPN(A2)
        POP     A5
        RTN
SAL:
        RTN
;
RENAME: ASCIZ   /CDR0:/              ; Nombre del CDROM
        EVEN
;
NOMEM:  EXIT
;
END
```

```
;**************************************
;**                                  **
;**    SUBRUTINA PARA ABRIR UN       **
;**      ARCHIVO DESDE BASIC         **
;**                                  **
;**  Nombre: XOPEN                   **
;**  Autor:  Jose Luis Damian G.     **
;**          Ricardo Zaldivar G.     **
;**  Fecha:  10/agosto/89            **
;**                                  **
;**************************************


; SUBRUTINA PARA ABRIR UN ARCHIVO EN EL CDROM PARA  PODER  USARLO
; EN BASIC

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG
    OBJNAM    .SBR                  ; Cambia la extension .LIT  por
                                    ; .SBR

    EXTERN    $FLSET

    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;
.OFINI
.OFDEF   BAS.DDB,D.DDB            ; DDB CONTROL BLOCK
.OFSIZ   BAS.SIZ

.OFINI
.OFDEF   CDR.DDB,D.DDB            ; DDB CONTROL BLOCK
.OFDEF   CDR.BUF,512.             ; Buffer  para  el   DDB   del
                                  ; DIRECT.CDR
.OFSIZ   CDR.SIZ
;
;Generamos el offset  para las varibles

.OFINI
.OFDEF   ARG.XC,2                 ; Offset   contador   de
                                  ; argumentos
.OFDEF   ARG.TYP,2                ; Offset de tipo
.OFDEF   ADD.XC,4                 ; Direccion de argumento
.OFDEF   SIZ.XC,4                 ; Tama&o del argumento
```

```
.OFINI
.OFDEF    FIL.NAM,12.                ; Nombre del archivo
.OFDEF    FIL.CHN,2                  ; Numero de canal
.OFSIZ    ARG.SIZ

:Generamos offset para el record de directorio

.OFINI
.OFDEF    NAM.FIL,12.                ; Nombre de Archivo
.OFDEF    BLK.FIL,4                  ; # Bloques en el Archivo
.OFDEF    BLK.INI,4                  ; Bloque inicial

START:    PHDR -1,0,PH$REE!PH$REU

; A0 contiene la direccion del inicio del area impura
; A1 registro para indexar el DDB del archivo abierto en basic
; A2 indice temporal, y apunta al DDB del archivo  abierto  en
;    BASIC
; A3 contiene la direccion de los argumentos
; A4 apunta a la base de la memoria libre
; A5 apunta al final de la memoria libre,
; A6 registro de scratch

; Verificacion de parametro y memoria disponible

    MOV       A5,D4
    SUB       A4,D4                  ; A5-A4
    CMP       D4,#CDR.SIZ
    BGT       0$
    TYPECR    ‹No hay  memoria  suficiente  para  usar  el  XCALL
XOPEN›
    EXIT
0$:
    MOV       A4,A1                  ; A1 guarda direccion  del  DDB
                                     ; para el DIRECT.CDR
    ADD       #CDR.SIZ,A4            ; Actualiza  la  base  de  la
                                     ; memoria libre en A4
    PUSH      A5                     ; Guardamos  la  direccion  del
                                     ; fin de la memoria
1$:
    TSTW      ARG.XC(A3)             ; Hay argumentos ?
    BNE       2$                     ; Si continua
    TYPECR    ‹Faltan argumentos en el XCALL XOPEN›
    EXIT

2$:
    CMPW      ARG.TYP(A3),#0
    BEQ       3$                     ; Es correcto continua
    TYPECR    ‹Tipo de variable invalido para el  uso  del  XCALL
XOPEN›
    EXIT
```

```
3$:
    CMP         SIZ.XC(A3),ARG.SIZ   ; Tamano de la variable
    BEQ         4$                   ; Es correcto continua
    TYPECR      ‹Tama&o de la variable invalido en el XCALL XOPEN›
    EXIT
4$:
    MOV         ADD.XC(A3),A5        ; Indexa    direccion     de
                                     ; argumentos
    CLR         D1
    MOVW        FIL.CHN(A5),D1
    CALL        $FLSET
    BEQ         5$                   ; Canal Correcto, continua
    TYPECR    ‹Se  paso  al  XCALL  XOPEN  un  canal  de   archivo
incorrecto›
    EXIT

5$:
    PUSH        A0                   ; Guarda  direccion  del  area
                                     ; impura y  libera  A0
    PUSH        A2                   ; Guardamos direccion del   DDB
                                     ; del Basic

INICIO:

;Busqueda de los datos del archivo que se desea abrir en el CDROM
; desde BASIC

;A0 apunta al nombre del arhivo  del  CDROM  especificado  en  el
; comando
;A2 sirve para indexar la FSPEC, y  para  indexar  el  buffer  de
; DIRECT.CDR
;D0 es registro  temporal  para  pasar  valores  al  DDB  y  como
; contador
;D1 se utiliza como respaldo de A2 cuando funciona como el indice
; del buffer
;D2 se utiliza para pasar  el  No.  de  registro  a  leer  en  el
; DIRECT.CDR
;D3 se utiliza como contador de registros en el buffer

    LEA         A2,NAME              ; Indexa archivo de directorio
    FSPEC       CDR.DDB(A1)          ; Pasa el nombre al DDB
; Inicializamos el DDB que generamos
    LEA         A2,CDR.BUF(A1)       ; A2   indexa    buffer    de
                                     ; DIRECT.CDR
    MOV         A2,D.BUF(A1)         ; Fija direccion del buffer
    MOV         #512.,D.SIZ(A1)      ; Fijamos tama&o del buffer
    MOVW        #20.,D.RSZ(A1)       ; Fija el tama&o del registro a
                                     ; leer
    MOVB        #‹D$INI!D$ERC!D$BYP›,D.FLG(A1)
                                     ; Fijamos banderas
    ANDB        #^C‹D$BYP›,D.FLG(A1); Habilitamos   mensajes   de
                                     ; error
    CLR         D2
    OPENR       CDR.DDB(A1),F.WAT    ; Abrimos DIRECT.CDR
```

```
12$:
    MOV         D2,D.REC(A1)        ; Fija el numero de record a
                                    ; leer
    INPUT       CDR.DDB(A1)         ; Leemos DIRECT.CDR
    LEA         A0,FIL.NAM(A5)      ; Indexa nombre del archivo
    MOV         #25.,D3             ; Fija contador de registros de
                                    ; DIREC
    MOV         A2,D1               ; D1 sera usado para
                                    ; actualizar A2
    MOV         #1,D0               ; Fija contador de
                                    ; caracteres = 0
8$:
    CMPB        (A2),#0             ; Es el fin del DIRECT.CDR
    JEQ         6$                  ; Si,salta
    CMPB        (A2),#'.            ; Se encontro el punto
    JNE         16$                 ; No, continua verificando
                                    ; nombre
; se inicia verificacion de extension
    CMP         D0,#8.              ; Es el numero de caracteres
                                    ; del nombre <= que el permitido
    BGT         17$                 ; Si continua verificando
                                    ; extension
    TYPECR      <Nombre de archivo fuente ilegal>
    EXIT
17$:
    MOV         #2,D0               ; Fija contador de Char. para
                                    ; la EXT
    INC         A2
    INC         A0
18$:
    CMPB        (A2)+,(A0)+         ; Compara Extension
    BNE         22$                 ; Salta al siguiente
                                    ; registro
    DBF         D0,18$              ; Sigue verificando extension
    BR          21$                 ; Se encontro el registro

16$:
    CMPB        (A2)+,(A0)+         ; Compara Nombre
    BEQ         7$                  ; Salta si no es igual
22$:
    ADD         #20.,D1             ; Incrementa apuntador para el
                                    ; siguiente registro de
    MOV         D1,A2               ; directorio
15$:
    LEA         A0,FIL.NAM(A5)      ; Indexa nombre del
                                    ; archivo
    DBF         D3,8$
    ADD         #25,D2
    JMP         12$
7$:
    INC         D0                  ; Incrementa contador de
                                    ; caracteres
    JMP         8$
```

```
21$:
    CLOSE     CDR.DDB(A1)
    BR        9$
6$:
    TYPECR    (No se encontro el archivo en el directorio del
CDROM)
    EXIT
9$:                                 ; Continua
; fijamos parametros en el DDB del archivo abierto en basic
    POP       A1                    ; Pasamos el DDB de Basic a A1
    LEA       A2,RENAME
    FSPEC     BAS.DDB(A1)
    ANDB      #^C(D$BYP),D.FLG(A1); Habilitamos mensajes de
                                    ; error
    MOV       D1,A2                 ; Regresa direccion de
                                    ; paramentros del archivo
                                    ; localizado en DIRECT.CDR
    MOV       BLK.FIL(A2),D0        ; Pasa tama&o del archivo a D0
    SWAP      D0
    CLR       D7
    MOVW      D0,D7
    MOV       #9,D6
    LSR       D0,D6                 ; Hacemos division entre 512
                                    ; rotando
    ANDW      #1FF,D7               ; La division es exacta entre
                                    ; 512
    BEQ       11$                   ; Si, salta
    ADD       #1,D0                 ; No, suma uno al residuo
11$:
    MOV       D0,D.FSZ+BAS.DDB(A1); Fijamos el tama&o del
                                    ; archivo
    MOV       #-1,D.LSZ+BAS.DDB(A1)
                                    ; Bytes en el ultimo block -1
    MOV       BLK.INI(A2),D0
    SWAP      D0
    MOV       D0,D.BAS+BAS.DDB(A1); Block inicial
SAL:
    POP       A0
    POP       A5                    ; Restaura apuntadores de
                                    ; memoria para poder regresar
                                    ; al BASIC
    RTN

RENAME:
    ASCIZ     /CDR0:/               ; Nombre para respaldo
NAME:
    ASCIZ     /DIRECT.CDR/          ; Nombre del archivo de
                                    ; directorio
    EVEN
;
NOMEM:
    EXIT
```

```
;*********************************************
;*                                           *
;*  SUBRUTINA PARA LEER UN SECTOR FISICO     *
;*   DEL CDROM Y PASARLO A UNA VARIABLE      *
;*   LOCALIZADA EN LA MEMORIA POR BASIC      *
;*                                           *
;*  Nombre: XREADM                           *
;*  Autor:  Jose Luis Damian G.              *
;*          Ricardo Zaldivar G.              *
;*  Fecha:  1/agosto/89                      *
;*                                           *
;*********************************************


; SUBRUTINA PARA LEER UN SECTOR FISICO DEL CDROM Y PASARLO A  UNA
; VARIABLE EN BASIC

    SEARCH    SYS
    SEARCH    SYSSYM
    SEARCH    TRM
    ASMMSG
    OBJNAM   .SBR                   ; Cambia la extension .LIT  por
                                    ; .SBR


    VMAJOR    = 1
    VMINOR    = 0
    VSUB      = 2
    VEDIT     = 100.

    RADIX     16
;
.OFINI
.OFINI   CDR.DDB,D.DDB              ; DDB para accesar el CDROM
.OFSIZ   CDR.SIZ

:Definicion de offset para las varibles

.OFINI
.OFDEF   ARG.XC,2                   ; Offset contador de
                                    ; argumentos
.OFDEF   ARG.TYP,2                  ; Offset de tipo
.OFDEF   ADD.XC,4                   ; Direccion de argumento
.OFDEF   SIZ.XC,4                   ; Tama&o del argumento

.OFINI
.OFDEF   NUM.BLK,4                  ; Numero de bloques
.OFDEF   VAR1,2048.                 ; Variable para el resultado
.OFSIZ   ARG.SIZ
```

```
START:    PHDR -1,0,PH$REE!PH$REU

; A0 contiene la direccion deL inicio del area impura
; A1 contiene la direccion del DDB para el CDROM
; A2 indice temporal, y apunta a la variable en la que se regresa
; la inf.
; A3 contiene la direccion de los argumentos
; A4 apunta a la base de la memoria libre
; A5 apunta al final de la memoria libre, se ocupara para indexar
; el DDB
; A6 registro de scratch

: verificamos si hay memoria suficiente

    MOV       A5,D4
    SUB       A4,D4
    CMP       D4,#CDR.SIZ
    BGT       1$
    TYPECR    ‹No hay memoria suficiente›
    EXIT
1$:
    MOV       A4,A1
    SUB       #CDR.SIZ,A4

    PUSH      A5                    ; Guardamos la direccion del
                                    ; fin de
    TSTW      ARG.XC(A3)            ; Hay argumentos ?
    BNE       2$                    ; No, regresa
    TYPECR    ‹faltan argumentos = ›
    EXIT
2$:
    CMPW    ARG.TYP(A3),#0          ; El tipo de argumento es
                                    ; correcto
    BEQ     3$                      ; Si, continua
    TYPECR  ‹Tipo de variable invalido ›
    EXIT
3$:
    CMP       SIZ.XC(A3),#ARG.SIZ   ; Es apropiado el tama&o del
                                    ; argumento es correcto ?
    BEQ       4$                    ; Si, continua
    TYPECR    ‹Tamano de la varible invalido ›
    EXIT
4$:
    MOV       ADD.XC(A3),A5         ; Indexa direccion de
                                    ; argumentos
    LEA       A2,VAR1(A5)           ; Indexamos direccion de VAR1
    PUSH      A2                    ; Guarda la direccion de VAR1
INICIO:
:
    LEA       A2,RENAME             ; Indexa el dispositivo CDR0:
    FSPEC     CDR.DDB(A1)
```

```
; A2 indexa la variable
    MOV       NUM.BLK(A5),D0       ; Numero de bloque a leer
    SWAP      D0
    POP       A2
    MOV       A2,D.BUF(A1)         ; Compartimos buffer con la
                                   ; variable
    MOV       #2048.,D.SIZ(A1)     ; Fijamos tama&o del buffer
                                   ; para CD
    MOVB #<D$INI!D$ERC!D$BYP>,D.FLG(A1)
                                   ; Fijamos banderas
    MOVW      #140A,D1             ; renglon,columna
    TCRT
    MOVW      #<-1_8.>+11,D1       ; Fijar baja intensidad
    TCRT
    TYPESP    <Leyendo de bloque # >
    MOVW      #<-1_8.>+12,D1       ; Fijar alta intensidad
    TCRT
    MOV       D0,D.REC(A1)
    MOVW      #141E,D1             ; renglon,columna
    TCRT
    CLR       D1
    MOV       D0,D1
    DCVT      0,OT$TRM
    READ      CDR.DDB(A1)
    POP       A5
    MOVW      #0FF00,D1            ; Borra pantalla
    TCRT
    RTN
SAL:    EXIT
RENAME: ASCIZ     /CDR0:/          ; Nombre para respaldo
    EVEN

    END
```

```
!**********************************************
!**                                          **
!**  PROGRAMA PARA VERIFICAR EL FUNCIONAMIENTO **
!**        DE LA SUBRUTINA EXTERNA XREADC    **
!**                                          **
!**  Nombre: XREADC                          **
!**  Autor:  Jose Luis Damian G.             **
!**          Ricardo Zaldivar G.             **
!**  Fecha:  7/agosto/89                     **
!**                                          **
!**********************************************


! ESTE EL REGISTRO UTILIZADO PARA LA SUBRUTINA EXTERNA XREADC

!Argumentos necesarios para la utilizacion de la XCALL XREADC

      MAP1 ARGUMENTOS
          MAP2 NUMERO'BLOQUES,B,4,0
          MAP2 BLOQUE'INICIAL,B,4,0
          MAP2 CANAL'ARCHIVO,B,2,

!Variables necesarias en el programa

       MAP1 RESULTADO,F
       MAP1 ARCHIVO,S,6

! Se piden los datos necesarios para correr el programa

      INPUT "NOMBRE DEL ARCHIVO DE SALIDA : ";ARCHIVO
      IF ARCHIVO="" GOTO END

      INPUT "NUMERO DE BLOQUES A LEER : ";NUMERO'BLOQUES
      INPUT "BLOQUE EN QUE SE INICIA LA LECTURA : ";BLOQUE'INICIAL

      LOOKUP ARCHIVO+".ESP",RESULTADO
      IF RESULTADO = 0 GOSUB CREA'ARCHIVO

      OPEN #1,ARCHIVO+".ESP",RANDOM,512,REC'NUM
      CANAL'ARCHIVO=1

!Verificamos los argumentos con los que entramos a la subrutina

      ?TAB(-1,0);TAB(2,1);"VOY A LA SUBRUTINA"
      ?"NUMERO DE BLOQUES = ";NUMERO'BLOQUES
      ?"BLOQUES INICIAL  = ";BLOQUE'INICIAL
      ?"CANAL ARCHIVO = ";CANAL'ARCHIVO

      INPUT "": NADA                        !Detiene ejecucion
```

```
      XCALL XREADC.ARGUMENTOS

      ?TAB(3,1):"REGRESE DE LA SUBRUTINA":
      CLOSE #1

END:
      END

CREA´ARCHIVO:

      ?TAB(-1,0);TAB(1,1);"VOY A CREAR EL ARCHIVO"; :   &
          INPUT "":NADA :          &
             ALLOCATE ARCHIVO+".ESP",NUMERO´BLOQUES
     RETURN
```

```
!*********************************************
!*                                           *
!*  PROGRAMA PARA PROBAR EL FUNCIONAMIENTO   *
!*      DE LA SUBRUTINA EXTERNA XOPEN        *
!*                                           *
!*   Nombre: XOPEN                           *
!*   Autor:  Jose Luis Damian G.             *
!*           Ricardo Zaldivar G.             *
!*   Fecha:  10/agosto/89                    *
!*                                           *
!*********************************************


!PROGRAMA DE PRUEBA PARA LA SUBRUTINA DE XOPEN

! Argumentos necesarios para utilizar la XCALL XOPEN

     MAP1 ARGUMENTOS
          MAP2 NOMBRE'ARCHIVO,S,12
          MAP2 CANAL'ARCHIVO,B,2


! Variables utilizadas en el programa

     MAP1 ENTRADA,S,1
     MAP1 ARCHIVO'PASO,S,12
     MAP1 REC'NUM,F,6,0
     MAP1 RESULTADO,F
     MAP1 VARIABLE'SALIDA,X,512
     ?TAB(-1,0)
     INPUT "CUAL ES EL ARCHIVO DE PASO: ": ARCHIVO'PASO
     INPUT "CUAL ES EL ARCHIVO QUE DESEAS ABRIR: ":  &
           NOMBRE'ARCHIVO

     NOMBRE'ARCHIVO=UCS(NOMBRE'ARCHIVO)

     IF NOMBRE'ARCHIVO="" OR ARCHIVO'PASO="" THEN GOTO END

     LOOKUP ARCHIVO'PASO,RESULTADO
     IF RESULTADO < 0 GOTO CON1
     ALLOCATE ARCHIVO'PASO,1

CON1:

     OPEN #1,ARCHIVO'PASO,RANDOM,512,REC'NUM
     CANAL'ARCHIVO=1

     ?TAB(5,7);"VOY A LA SUBRUTINA": : INPUT "": NADA
     XCALL XOPEN,ARGUMENTOS
     ?TAB(7,7);"REGRESO DE LA SUBRUTINA": : INPUT "": NADA
```

```
! Al regresar de la subrutina, el sistema ya debe estar
! capacitado para accesar en forma logica el archivo indicado en
! el CDROM

     INPUT "CUANTOS BLOCKS QUIERES LEER":N
     FOR I=1 TO N

! Lee el archivo en el CDROM

     READ #1,VARIABLE'SALIDA

     ?TAB(-1,0);TAB(1,1);VARIABLE'SALIDA
     REC'NUM=REC'NUM+1
     NEXT I

! El archivo se puede cerrar en forma ! convecional

     CLOSE #1

END:

     ?TAB(24,1);"TERMINE DE EJECUTAR EL PROGRAMA"
     END
```

```
**********************************************
**                                          **
**  ESTE PROGRAMA UTILIZA LA SUBRUTINA EXTERNA  **
**   XREADM Y FUE UTILIZADA PARA ENCONTRAR EL   **
**      DIRECTORIO DE DEL CDROM DE LA UNAM      **
**                                          **
**  Nombre: XREADM                          **
**  Autor:  Jose Luis Damian G.             **
**          Ricardo Zaldivar G.             **
**  Fecha:  1/agosto/89                     **
**                                          **
**********************************************


! ESTE PROGRAMA ENCUENTRA INFORMACION ESPECIFICA EN CUALQUIER
! SECTOR DEL CDROM HACIENDO USO DE LA SUBRUTINA EXTERNA XREADM

! Variable necesaria para utilizar la subrutina externa XREADM
    MAP1 XC'VARIABLE
         MAP2 NUMERO'BLOCK,B,4,0
         MAP2 VAR1,X,2048,0
!
! Variables necesarias en el programa
    MAP1 RESULTADO,F
    MAP1 ARCHIVO,S,6
    MAP1 POSICION(10),B,2
    MAP1 OFFSET,B,2,1
    MAP1 NUM'SEC,B,4,0

! Se piden los datos necesarios para la ejecucion del programa

    ?TAB(-1,0);TAB(5,10);"NOMBRE DEL ARCHIVO DE SALIDA DE ";   &
         "RESULTADOS  ";
    INPUT "";ARCHIVO

    IF ARCHIVO="" THEN ?TAB(-1,0);TAB(1,1);"Error en la ";  &
         "especificacion del archivo de salida": GOTO END

    ?TAB(7,10);"Numero de Sectores a leer  ";
    INPUT "";NUM'SEC

    ?TAB(9,10);"En que bloque quieres iniciar ";
    INPUT "";NUMERO'BLOCK

    LOOKUP ARCHIVO+".CDR",RESULTADO
    IF RESULTADO <>0 THEN GOTO CONTINUA

    OPEN #100,ARCHIVO+".CDR",OUTPUT
    CLOSE #100
```

```
CONTINUA:

! Se fijan los datos que se desean buscar en el CDROM

        A$(1)="DATFIJO1"
        A$(2)="DATVARI1"
        A$(3)="LLAAUTR1"
        A$(4)="LLAGENE1"
        A$(5)="LLATEMA1"
        A$(6)="LLATITR1"
        A$(7)="RECGENE1"
        A$(8)="RECTEMA1"
        A$(9)="RESAUTO1"
        A$(10)="RESTITU1"


   FOR I=1 TO NUM'SEC

        XCALL XREADM,XC'VARIABLE

        OPEN #100,ARCHIVO+".CDR",APPEND

        FOR J=1 TO 10

BUSCA'MAS:

            POSICION(J)=INSTR(OFFSET,VAR1,A$(J))
            OFFSET=POSICION(J)+LEN(A$(J))

            IF POSICION(J)=0 THEN  OFFSET=1 : NEXT J :  &
                 CLOSE #100 : NUMERO'BLOCK=NUMERO'BLOCK+4 : &
                    NEXT I : GOTO END

            ? #100: "POSICION DEL STRING = ";J; "  en  ";     &
                 POSICION(J): "  En el block "; NUMERO'BLOCK

            IF (LEN(VAR1)-POSICION(J)) > LEN(A$(J))     *
                  THEN GOTO BUSCA'MAS
        NEXT J
   NEXT I

END:

     END
```

5.4 PRUEBAS E INTEGRACION

Las pruebas son un proceso por el que todos los programas pasan.

Las pruebas, se pueden definir como la ejecución de un programa, bajo ciertas condiciones, con objeto de encontrar posibles errores.

Existen diferentes tipos de pruebas para verificar que los programas hacen lo esperado desde el análisis y el diseño. Entre los tipos de prueba podemos distinguir los siguientes:

I. Las pruebas de " Caja Negra ", que nos permiten diseñar los casos de prueba en función de las especificaciones del programa, sin tener que tomar en cuenta la lógica interna. Es decir, que se detectan errores con solo determinar si las salidas son las esperadas de acuerdo a las entradas.

II. Las pruebas de " Caja Blanca o Lógicas ", que nos permiten diseñar los casos de prueba de acuerdo a la estructura interna del programa. Es decir, se definen entradas para que el programa realice ciertas instrucciones, pase por ciertas condiciones, etc.

De acuerdo a lo anterior, podemos decir, que los tipos de pruebas que se hacen a los programas, dependen de las características de los mismos.

En general, las primeras pruebas que se efectuan, son del tipo de caja negra y dependiendo del resultado de éstas, se diseñan pruebas de caja blanca.

Como ya habíamos mencionado, cada proceso requiere de un tipo de pruebas especiales, las cuales a continuación trataremos de describir para nuestro caso particular.

Antes que nada, se requieren de los elementos a probar, concretamente nos referimos a tener disponibles CD-ROMs con información en formato High Sierra, de preferencia más de un disco para pruebas.

También es importante tener parámetros de comparación. Para esto, podemos decir que la mayoría de los CD-ROMs vienen con cierta documentación sobre su uso y la información que tienen.

En este caso, los elementos necesarios, han sido proporcionados por la Dirección General de Bibliotecas de la UNAM (DGB), los cuales fueron obtenidos de un sistema para CD-ROM que funciona actualmente con las PCs.

A continuación definiremos algunas de las pruebas para los procesos de este proyecto:

I. El primer proceso a probar es el referente al driver del CD-ROM.

   a) Se crean diferentes drivers con diferentes características: con formato extendido o tradicional; diferentes números de subsistemas: con diferentes versiones de sistema operativo; etc.

   b) Se instala el CD-ROm con cada uno de los drivers generados.

   c) Se trata de leer la información del CD-ROM por medio de los comandos que lean en forma física (ASCDMP o DUMP).

   d) Se verifican los resultados con los parámetros de comparación (con la información proporcionada del CD-ROM).

II. Pruebas de los Comandos.

   Las pruebas a estos tipos de programas, son básicamente pruebas de caja negra. Esto es principalmente a que son procesos simples de una sola tarea.

   La forma en que se prueban estos procesos y los resultados esperados, se presentan a continuación:

   a) Se teclea DIRCD ‹RET› y debe aparecer en la pantalla de la terminal, el directorio del disco.

   b) Se teclea LABCD ‹RET› y debe aparecer en la pantalla de la terminal la etiqueta del disco.

   c) Se teclea COPYCD destino=archivo-fuente-en-el-CDROM y debe generar un archivo en el disco magnético que contenga la información del archivo del CD-ROM.

   d) Se teclea TYPECD nombre ‹RET› y debe aparecer en pantalla el contenido del archivo especificado.

En caso de que las pruebas de caja negra no den el resultado esperado, se prueba el programa con el debbuger FIX instrucción por instrucción hasta encontrar el error.

FIX nombre-programa

III. Prueba de la Rutinas.

Primero de prueban por separado con FIX.

a) FIX XREADC.SBR

b) FIX XOPEN.SBR

c) FIX XREADM.SBR

Estando en FIX se ejecuta el programa en Basic para que coloque los parámetros adecuados, y luego se corre la rutina, ya sea completa o instrucción por instrucción.

d) GO RUN XREADC

e) GO RUN XOPEN

f) GO RUN XREADM

Una vez que la rutina arroja los resultados deseados, se prueba llamándolo directamente al correr el programa Basic.

g) RUN XREADC

h) RUN XOPEN

i) RUN XREADM

En nuestro caso, los programas para probar las subrutinas, tiene el mismo nombre que las subrutinas que van a probar.

IV. Prueba de los Programas de Basic.

Los programas de Basic, que para este proyecto se requieren son fáciles y sencillos, por lo que con pruebas de caja negra se detectan los errores.

Una vez que ya estan probados por separado, todos y cada uno de los procesos que conforman al sistema lector de CD-ROM, viene la etapa de integración.

También para la integración existen diferentes formas de hacerla, así tenemos integración incremental top-down, bottom-up, sandwich, etc.

Sin embargo en nuestra caso la integración es casi nula, ya que cada proceso y programa en su mayoría es idependiente sobre todo los que se manejan como comandos.

A pesar de lo anterior los procesos elementales sí se integran a otros procesos. Por ejemplo, el proceso de acceso físico al CD-ROM es un proceso independiente pero que se integra a la mayoría de los demás procesos, ya que de algún modo hacen referencia a lecturas físicas. También el proceso de obtención de directorio, se integra a procesos que requieran identificar un archivo en particular del CD-ROM, sin información previa alguna, por ejemplo la abertura, despliegue y copiado de archivos.

Con los programas que son rutinas, la integración se realiza en el momento en que dicha rutina se integra al programa en Basic.

5.5 RESULTADOS

Los resultados de este proyecto fueron satisfactorios, dado que se llego a cumplir con los principales objetivos planteados, entre los cuales el más importante fué el de desarrollar un sistema lector de CD-ROM para un sistema basado en el microprocesador 68000.

En este momento se cuenta con un conjunto de elementos que permiten accesar y manipular en forma general y desde un lenguaje en alto nivel la información de cualquier CD-ROM.

Ahora bien, los resultados concretos obtenidos durante el desarrollo del sistema, se presentan a continuación tratando de seguir la secuencia de desarrollo.

Respecto a la conexión física, podemos decir que se realiza en una forma muy sencilla, ya que solo se necesita contar con los elementos necesarios (lo cual ya se describio anteriormente) y ésta se lleva a cabo en unos pocos minutos.

Antes de lograr una conexión lógica satisfactoria, nos referimos con esto a la obtención del driver del CD-ROM; se probaron los diferentes drivers que se pueden generar. Llegando a la conclusión de que los drivers generados con la utilerías que nos proporciona el sistema, no funcionan con versiones del sistema operativo anteriores a la 2.0, y que el formato de disco con el que se obtienen mejores resultados es con el extendido, ya

que, haciendo uso del formato tradicional con la versión 2.0 se tienen algunos problemas para el manejo de la información del CDROM debido a que los drivers limitan el tamaño de los discos lógicos a 32 MB y en muchos casos, los archivos grabados pueden quedar dentro de unidades lógicas diferentes lo cual dificulta el manejo de los mismos. Por otro lado, los drivers simulan los bloques 0, 1 y 2 de cada unidad lógica provocando que no se pueda accesar información valida en el CDROM, lo que ocasiona que el uso del formato tradicional sea descartado para el manejo del CDROM.

Adicionalmente, la forma de manejo del CDROM que sugiere ALPHA MICRO que es la de generar el area 1,2 en el CDRO y crear el archivo CDROM.DAT utilizando cualesquiera de los dos formatos de disco, tampoco se puede utilizar con versiones de sistema operativo posteriores o iguales a la 2.0, por lo que, la forma de manejo del CDROM que nosotros estamos haciendo es completamente nueva, y de creación propia.

La verificación de los procesos de acceso fisico, fueron satisfactorios, ya que contamos con un disco del cual sabíamos en que posicion se encontraba la información que contenía, pudiendo con esto, verificar si la información que estabamos leyendo era correcta o no. Esto se pudo llevar a cabo en forma satisfactoria gracias a la ayuda que nos proporciono el Departamento de Informática de la DGB en el sentido de verificar si la información que nosotros obteniamos era correcta, y por otro lado, proporcionandonos algunos datos referentes a la información que contiene su CDROM (por ejemplo: el nombre, tamaño, formato y número de archivos), que fueron de gran utilidad para nosotros.

Para la obtención del directorio real del CDROM, se tuvieron problemas debido a que no contamos con información suficiente sobre la estructura con la que está hecho. Por tal motivo, nos vimos en la necesidad de elaborar un directorio exclusivo del CDROM que estamos usando, para con esto poder probar todos los programas o subrutinas que hacen una manipulación lógica de la información como el TYPECD, COPYCD o el XOPEN. Por lo cual, en estos momentos solo se puede accesar en forma lógica el CDROM que estamos usando para el desarrollo. Sin embargo, cualquier disco CDROM se puede accesar en estos momentos en forma fisica, lo cual, nos permitirá trabajar en forma rapida con otros discos.

Por el motivo antes mencionados, todos los programas que hacen uso del directorio del CDROM, se dejaron preparados para que en el momento en que ya se pueda tener acceso al directorio real del CDROM se puedan corregir en una forma sencilla, y de esta forma se pueda tener acceso a la información de cualquier disco CDROM.

En estos momentos el proceso de desarrollo se encuentra detenido por falta de información de la estructura del directorio y por falta de más discos con Formato High Sierra, que nos permitan comparar la estructura de los discos, para poder identificar plenamente la localización de algunas estructuras importantes de los discos como son: La raíz del directorio, la etiqueta del disco, etc..

Hasta el momento el CDROM se pude utilizar en forma satisfactoria con el equipo de cómputo con el que contamos, y en primera instancia se podría dar inicio al desarrollo de software para procesar la información de la DGB, y con esto comenzar a aprovechar el trabajo que realizamos. No obstante, cuando se realizen las modificaciones necesarias a los programas que se elaboraron para hacerlos de uso general de cualquier CDROM, el software que se haya hecho para la manipulación de la información de la DGB no sufrirá ningún cambio, ya que se conservará el mismo funcionamiento de las subrutinas y comandos que actualmente se les proporcionan.

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

Al desarrollar el sistema lector de CD-ROM, para un sistema con microprocesador 68000, pudimos apreciar que los conocimientos adquiridos durante el estudio de la carrera de ingeniería fueron los adecuados para poder atacar este problema, ya que por un lado, fueron lo suficientemente generales, para poder comprender lo que se quería del sistema y conocer las posibilidades de llevarlo a cabo. Y por otro lado, los estudios fueron lo sufientemente particulares. coma para poder aprender por medio de analogías lo que no sabíamos.

Nos referimos a los conocimientos generales, básicamente al uso de metodologías y técnicas de desarrollo, las cuales nos permiten entender el problema, plantear alternativas de solución y planear.

A los conocimientos particulares, no referimos por ejemplo, al hecho de conocer lenguajes de programación como ensamblador Z80 que nos ayudo a entender el ensamblador 68000, lenguaje Basic, compiladores, ligadores, debbugers, etc.. Estos conocimientos nos facilitaron el desarrollo, ya que redujeron el tiempo de aprendizaje del ensamblador 68000 y del AlphaBasic, ya que por medio de analógias o inferencias pudimos asimilarlos con facilidad.

Adicionalmente, los conocimientos generales nos ayudaron a enteder el funcionamiento de todo el sistema de cómputo que utilizamos, lo cual redundo también en minimizar el tiempo de desarrollo, ya que de no haberlos tenido este tiempo hubiera sido considerablemente más largo, tomando en cuenta que se trata de un tema muy nuevo.

En esta tesis, se desarrolló un sistema que nos permitió aplicar, en forma general, los conocimientos de electrónica adquiridos durante el estudio de la carrera, concretamente nos referimos a la electrónica digital. También podemos apreciar que el desarrollo de este sistema tiene mucho que ver con lo que se refiere a técnicas, procedimientos y herramientas de programación de computadoras, por lo que a este respecto tuvimos que documentarnos, consultar y aprender muchas de las cosas que para los profesionales de la computación son obvias. Nos referimos por ejemplo a la Programación Estructurada.

Por otra parte, se cumplió el objetivo de esta tesis, ya que se obtuvo un sistema lector de CD-ROM, aunque por el momento es exclusivamente para la DGB, no abstante, se puede generalizar para trabajar con cualquier disco CD-ROM con el Formato High Sierra, con cambios pequeños , una vez que se tenga la información necesaria sobre la estructura del directorio del CD-ROM. Todas las nuevas modificaciones se considerarán como nuevas etapas de desarrollo en las cuales se puede mejorar, ampliar o particularizar el sistema según las necesidades de los usuarios una vez que éstos comiencen a utilizar el sistema.

Entre las modificaciones que nosotros tenemos contempladas para el sistema se encuentran las siguientes:

- Utilización del directorio real del disco.
- Adicionar simbolos wildcar en el comando DIRCD, lo cual facilitará la busqueda de archivos.
- Desarrollo de aplicaciones generales en el CD-ROM.
- Desarrollo de aplicaciones especificas, las cuales, estarán sujetas a las necesidades de cada usuario.
- Implementación de nuevas subrutinas según las necesidades que se detecten en los usuarios.

El sistema lector de CD-ROM, en las condiciones en que se encuentra actualmente, ya puede ser utilizado por la DGB para la elaboración de un sistema multiusuario que pueda hacer uso del disco CD-ROM que mando hacer la propia DGB y que contiene toda la información bibliográfica de la misma. Este sistema agilizará el proceso de consulta o búsqueda de información ya que actualmente ésto se realiza mediante el uso de una PC, las cuales tienen la restricción de que por cada drive de CD-ROM debe existir un PC. Ahora con el sistema mutiusuario con un solo drive de CD-ROM, varios usuarios podrán consultar el CD-ROM al mismo tiempo.

Una de las cosas que pudimos apreciar, fué la difucultad para realizar investigaciones en México, ya que se cuenta con poca información, disponible en el país, sobre los avances tecnológicos recientes, como lo es en nuestro caso el CD-ROM. Por otro lado, existen personas que obstaculizan las investigaciones, o no quieren compartir los conocimientos o la información que ellos poseen sobre el tema, temiendo tal vez que se vean afectados sus intereses, tal fue el caso que nosotros tuvimos con la compañía que hizo el CD-ROM para la UNAM, la cual, no quiso facilitarnos ninguna información sobre el Formato High Sierra, que es con el que se supone ellos fabricaron el CD-ROM.

Por otro lado, cabe mencionar que hay muchas personas que se encuentran trabajando actualmente con los discos CD-ROM pero que sin embargo, no tienen información suficiente sobre ellos. Lo cual es una evidencia clara de la dependencia que todavía tenemos en México sobre las nuevas tecnológias ya que las utilizamos sin dominarlas completamente.

En nuestro caso nosotros estamos dependientes de Estados Unidos porque es ahí en donde se encuentra la información sobre la Propuesta del Formato High Sierra, la cual nosotros pedimos poco tiempo después de que empezamos a trabajar con el CD-ROM, y actualmente todavía no nos la mandan. Por lo cual, nos vemos en la necesidad de dar por terminado el desarrollo del sistema lector de discos ópticos CD-ROM en lo que concierne a nuestro trabajo de tesis. Sin embargo, esto no quiere decir que no se va a seguir trabajando en el desarollo del sistema, ya que por el momento, se puede continuar trabajando junto con la DGB en el desarrollo del software que necesita para el manejo de la información de su CD-ROM, y en cuanto se tenga la información que necesitamos del Formato High Sierra se harán las modificaciones que se necesitan para hacer generales los programas desarrollados.

APENDICES

A P E N D I C E A

CONJUNTO DE INSTRUCCIONES DEL MICROPROCESADOR 68000

Mnemonic	Description	Operation	Condition codes				
			X	N	Z	V	C
ABCD	Add Decimal with Extend	(Dst)10 + (Fnt)10 + X --> Dst	*	U	*	U	*
ADD	Add Binary	(Dst) + (Fnt) --> Dst	*	*	*	*	*
ADDA	Add Addres	(Dst) + (Fnt) --> Dst	-	-	-	-	-
ADDI	Add Inmediate	(Dst) + Inmediate Data --> Dst	*	*	*	*	*
ADDQ	Add Quik	(Dst) + Inmediate Data --> Dst	*	*	*	*	*
ADDX	Add Extended	(Dst) + (Fnt) + X --> Dst	*	*	*	*	*
AND	AND Logical	(Dst) & (Fnt) --> Dst	-	*	*	0	0
ANDI	AND Inmediate	(Dst) & Inmediate Data --> Dst	-	*	*	0	0
ANDI to CCR	AND Inmediate to Condition Codes	(Fnt) & CCR --> CCR	*	*	*	*	*
ANDI to SR	AND Inmediate to Status Register	(Fnt) & SR --> SR	*	*	*	*	*
ASL, ASR	Arithmetic Shift	(Dst) Shifted by (count) --> Dst	*	*	*	*	*
Bcc	Branch Conditionally	If cc then PC + d --> PC	-	-	-	-	-
BCHG	Test a Bit and Change	~(<bit number>) OF Destination --> Z ~(<bit number>) OF Destination --> <bit number> OF Destination	-	-	*	-	-
BCLR	Test a Bit and Clear	0--><bit number>--> OF Destination ~(<bit number>) OF Destination --> Z	-	-	*	-	-
BRA	Branch Always	PC + d --> PC	-	-	-	-	-

Mnemonic	Description	Operation	Condition codes				
			X	N	Z	V	C
BSET	Test a Bit and Set	1-->(bit number) OF Destination ~(bit number) OF Destination --> Z	-	-	*	-	-
BSR	Branch to Subrutina	PC --> -(SP); PC + d --> PC	-	-	-	-	-
BTST	Test a Bit	~(bit number) OF Destination --> Z	-	-	*	-	-
CHK	Check Register Against Bounds	If Dn<0 or Dn>(ea)) then TRAP	-	*	U	U	U
CLR	Clear and Operand	0-->Destination	-	0	1	0	0
CMP	Compare	(Dst) - (Fnt)	-	*	*	*	*
CMPA	Compare Address	(Dst) - (Fnt)	-	*	*	*	*
CMPI	Compare Immediate	(Dst) - Immediate Data	-	*	*	*	*
CMPM	Compare Memory	(Dst) - (Fnt)	-	*	*	*	*
DBcc	Test Condition, Decrement and Branch	If ~cc then Dn-1-->Dn; if Dn-1 then PC+d-->PC	-	-	-	-	-
DIVS	Signed Divide	(Dst)/(Fnt) --> Dst	-	*	*	*	0
DIVU	Unsigned Divide	(Dst)/(Fnt) --> Dst	-	*	*	*	0
EOR	Exclusive OR Logical	(Dst)⊕(Fnt) --> Dst	-	*	*	0	0
EORI	Exclusive OR Immediate	(Dst)⊕Immediate Data --> Dst	-	*	*	0	0
EORI to CCR	EORI to Condition Codes	(Fnt)⊕CCR --> CCR	*	*	*	*	*
EORI to SR	EORI to Status Register	(Fnt)⊕SR --> SR	*	*	*	*	*
EXG	Exchange Register	Rx<-->Ry	-	-	-	-	-
EXT	Sign Extend	(Dst)SignExtend --> Dst	-	*	*	0	0
JMP	Jump	Dst --> PC	-	-	-	-	-
JSR	Jump to Subroutine	PC --> -(SP); Dst --> PC	-	-	-	-	-
LEA	Load Efective Address	(ea) --> An	-	-	-	-	-
LINK	Link and Allocate	An --> -(SP); SP --> An; SP+Displacement --> SP	-	-	-	-	-
LSL, LSR	Logical Shift	(Dst) Shifted by (count) --> Dst	*	*	*	0	*
MOVE	Move Data from Source to Destination	(Fnt) --> (Dst)	-	*	*	0	0

Mnemonic	Description	Operation	Condition codes				
			X	N	Z	V	C
MOVE to CCR	Move to Condition Code	(Fnt) --› CCR	*	*	*	*	*
MOVE to SR	Move to Status Register	(Fnt) --› SR	*	*	*	*	*
MOVE from SR	Move from the Status Register	SR --› Dst	-	-	-	-	-
MOVE USP	Move User Stack Pointer	USP --› An; An --› USP	-	-	-	-	-
MOVEA	Move Address	(Fnt) --› Dst	-	-	-	-	-
MOVEM	Move Multiple Register	Register --› Dst (Fnt) --› Register	-	-	-	-	-
MOVEP	Move Peripheral Data	(Fnt) --› Dst	-	-	-	-	-
MOVEQ	Move Quick	Inmediate Data --› Dst	-	*	*	0	0
MULS	Signed Multiply	(Dst)x(Fnt) --› Dst	-	*	*	0	0
MULU	Unsigned Multiply	(Dst)x(Fnt) --› Dst	-	*	*	0	0
NBCD	Negate Decimal with Extend	0-(Dst)10-X--› Dst	*	U	*	U	*
NEG	Negate	0-(Dst) --› Dst	*	*	*	*	*
NEGI	Negate with Extend	0-(Dst)-X --› Dst	*	*	*	*	*
NOP	No Operation	-	-	-	-	-	-
NOT	Logical Complement	~(Dst) --› Dst	-	*	*	0	0
OR	Inclusive OR Logical	(Dst)v(Fnt) --› Dst	-	*	*	0	0
ORI	Inclusive OR Inmediate	(Dst) v Inmediate Data --› Dst	-	*	*	0	0
ORI to CCR	ORI to Condition Codes	(Fnt) v CCR ---› CCR	*	*	*	*	*
ORI to SR	ORI to Status Register	(Fte) v SR ---› SR	*	*	*	*	*
PEA	Push Effective Address	(ea) ---› -(SP)	-	-	-	-	-
RESET	Reset External Device	-	-	-	-	-	-
ROL,ROR	Rotate (Without Extend)	(Dst)Rotated by (count)---› Dst	-	*	*	0	*
ROXL,ROXR	Rotate With Extend	(Dst)Rotated by (count)---› Dst	*	*	*	0	*
RTE	Return from Exception	(SP)+ --› SR; (SP)+ --› PC	*	*	*	*	*

Mnemonic	Description	Operation	Condition codes				
			X	N	Z	V	C
RTE	Return and Restore Condition Codes	(SP)+ --› CC; (SP)+ --› PC	*	*	*	*	*
RTS	Return from Subrutina	(SP)+ --› PC	-	-	-	-	-
SBCD	Substract Decimal with Extend	(Dst)10-(Fnt)10-X --› Dst	*	U	*	U	*
Scc	Set According to Condition	If cc then 1's --› Dst else 0's --› Dst	-	-	-	-	-
STOP	Load Status Registeer and Stop	Inmediata Data --› SR; Stop	*	*	*	*	*
SUB	Substract Binary	(Dst)-(Fnt) --› Bst	*	*	*	*	*
SUBA	Substract Address	(Bst)-(Fnt) --› Dst	-	-	-	-	-
SUBI	Substract Inmediate	(Bst)-Inmediate data --› Dst	*	*	*	*	*
SUBQ	Substract Quick	(Bst)-Inmediate data --› Dst	*	*	*	*	*
SUBX	Substract with Extend	(Bst)-(Fnt)-X --› Dst	*	*	*	*	*
SWAP	Swap Register Halves	Register[31:16](--)Register[15:0]	-	*	*	0	0
TAS	Test and Set an Operand	(Dst) Tested --›cc; 1--› [7] OF Dst	-	*	*	0	0
TRAP	Trap	PC --› -(SSP); SR --› -(SSP); (Vector) --› PC	-	-	-	-	-
TRAPV	Trap on Overflow	If V then TRAP	-	-	-	-	-
TST	Test and Operand	(Bst) Tested --› cc	-	*	*	0	0
UNLK	Unlink	An --› SP; (SP)+ --› An	-	-	-	-	-

&	logical AND	*	affected
v	logical OR	-	unaffected
@	logical exclusive OR	0	cleared
¯	logical complement	1	set
[ ]#	bit number	U	undefined

A P E N D I C E B

CARACTERISTICAS TECNICAS DEL REPRODUCTOR DE CD-ROM

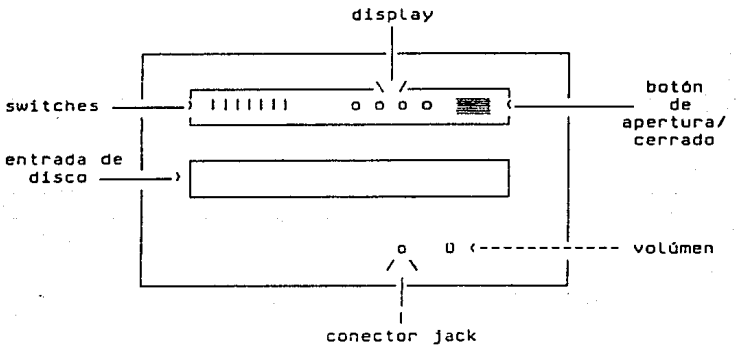
El drive para CD-ROM que utilizamos en este proyecto es un modelo TOSHIBA XM-2100A.

Las caracteristicas principales del drive son las siguientes:

1. Incluye una fuente de poder para poder proporcionar la energia requerida por el drive.
2. Equipado con un controlador de interfaz SCSI.
3. Conectores de entrada/salida que permiten la conexión "daisy chain".
4. Alta relación para utilizar el sistema CIRC de correción de errores. (Basado en el MODO 1 seleccionable).
5. 7 bloques de datos (16 Kbytes) para el buffer, con opción para expansión a 27 bloques (64 Kbytes). Esto permite la alta eficiencia en la transmisión de datos.
6. Acceso rápido (promedio máximo de 700 ms), en este influye el poderoso motor lineal y el nuevo sistema empleado para reconocimiento y búsqueda.
7. Función para reproducción de audio digital.
8. Carga automática frontal.

OPERACION, CONTROLES Y FUNCIONES

En la siguiente figura tenemos la parte frontal del drive que contiene los elementos que a continuación se describen:



1. Entrada para el disco.

   Es donde se coloca el disco, se abre y se cierra automáticamente al recibir la orden correspondiente de la computadora o al oprimir el botón que para este efecto se tiene.

2. Botón de Apertura/Cerrado.

   Abre y cierra la entrada para el disco. Para prevenir que se abra por accidente, para abrir es necesario tener oprimido el botón por 2 o 3 segundos.

3. Display.

   Son leds que indican lo siguiente:

   POWER. Se enciende la luz cuando se enciende el drive.

   DISC. Se enciende cuando el disco se carga correctamente.

BUSY. Se enciende cuado se accesa el disco o cuando hay transferencia de datos.

AUDIO. Se enciende cuando se esta reproduciendo audio.

4. Conector tipo "jack".

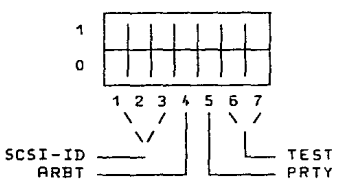
Acepta un conector tipo jack de 3.5 mm. para monitorear señales de audio que sean reproducidas.

5. Volúmen.

Ajusta el sonido del volúmen.

6. Panel de Switches.

Se tiene un conjunto de 7 switches que a continuación se describen:



SCSI-ID. Con los tres primeros switches se define el número de disco que se manejará con la interfaz SCSI. La definición se hace como número binario, teniendo la siguientes equivalencias.

SCSI-ID	SWITCH 1	2	3
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

PRTY. El switch 4, para verificar o no la paridad.

Switch	Función
0	El drive no verifica la paridad, pero manda los datos con ésta.
1	El drive verifica la paridad.

ARBT. Para habilitar o deshabilitar el sistema de arbitrio. Es el switch número 5.

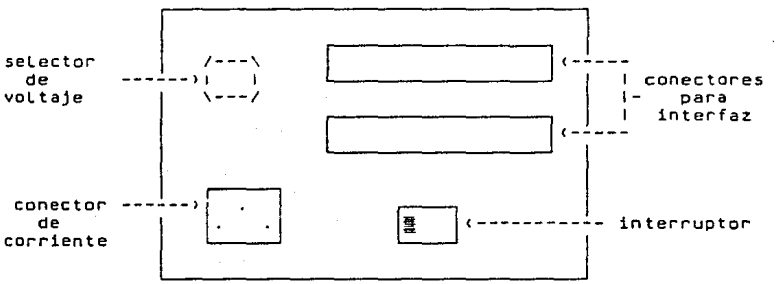
Switch	Función
0	Sin sistema de arbitrio
1	Con sistema de arbitrio

TEST. Los diferentes tipos de pruebas que el drive puede realizar, se definen por medio de los switches 6 y 7. En la siguiente tabla se muestran los valores de estos switches.

Cuando se efectua alguna prueba, los demás switches deben estar en la posición cero.

Switch	Función
00	Modo de Operación Normal
01	Modo 1 Ajuste de Foco
10	Modo 2 Ajuste de Pista
11	Modo de Reproducción de audio

En la siguiente figura tenemos la parte posterior del drive, que contiene los elementos que a continuación se describen:



7. Interruptor.

8. Conector de Corriente.

9. Selector de Voltaje.

   Selecciona uno de cuatro voltajes (100V, 120V, 220V y 240V).

   No se debe olvidar seleccionar primero el voltaje utilizado, antes de conectar el cable de corriente.

10. Conectores para la Interfaz.

    Utilizan conexión para el bus SCSI.

    Cuando no se utiliza el tipo de conexión "daisy chain", o el drive es el último de la cadena, el conector de salida debe terminar con un terminador SCSI.

TORNILLO FIJADOR

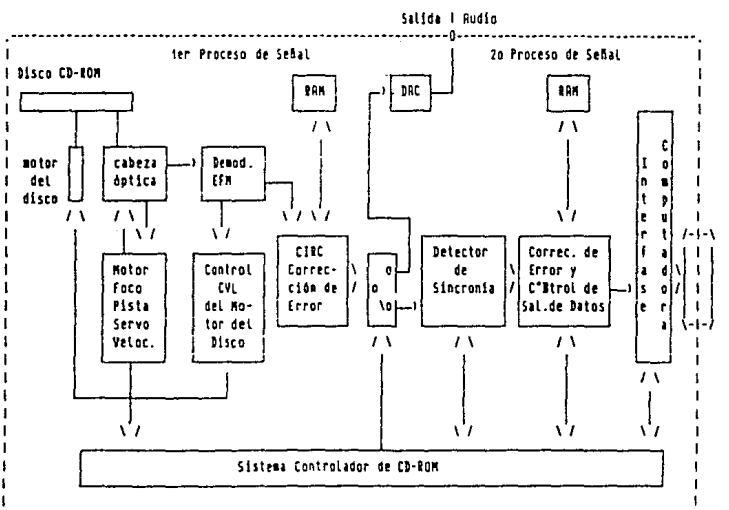
El drive, tiene en la parte inferior, un tornillo fijador de las cabezas del disco.

Este tornillo tiene como objetivo proteger las cabezas y el disco cuando el drive va a ser transportado.

Antes de utilizar el drive, el tornillo fijador de cabezas, se debe remover de su lugar original. Se puede colocar en un orificio que se encuentra tambien en la parte inferior del drive, para que éste no se pierda.

CONFIGURACION GENERAL DEL SISTEMA CD-ROM

En el siguiente dibujo se presenta la configuración general del sistema:



ESPECIFICACIONES GENERALES

Funcionamiento

1. Capacidad de Datos (de acuerdo a las especificaciones del CD-ROM, calculada en 65 minutos)

a)	Total de Datos del Usuario	Modo 1 Modo 2	599M bytes 683M bytes
b)	Total Número de Bloques		292.5K blocks
c)	Bloques/Datos Usuario	Modo 1 Modo 2	2048 B/D 2336 B/D

2. Velocidad de Transferencia.

a)	Promedio (Máxima vel. en lectura)	 Modo 1 Modo 2	75Blocks/s 153.6Kbyte/s 175.2Kbyte/s
b)	Máximo (del buffer)		1.4Mbyte/s

3. Tiempo de Acceso (incluyendo latencia)

a)	Tiempo de Acceso	400ms
b)	Máximo	menos de 700 ms

(valor promedio para más de 1000 veces de acceso al minúto 60, segundo 1, bloque 74 del track más interno)

4. Capacidad en el Buffer de Datos — 16Kbytes / Opción 64Kbytes

5. Errores de Lectura (El rango de error del disco debe ser 10^{-3}) — Modo 1: menos de 10^{-12}; Modo 2: menos de 10^{-9}

6. Cabeza Optica.

a)	Laser	Laser semiconductor
b)	Mecanismo de la cabeza	Sistema de motor lineal

7. Revolución — Aprox. 200~530 rpm (CLV)

Indicador de Potencia

1.	Fuente de Poder	AC 100V/120V/220V/240V 50V/50Hz ± 10%
2.	Indicador.	
	a) POWER	
	b) DISC	
	c) BUSY	
	d) AUDIO	
3.	Conector para la interfaz	INPUT/OUTPUT (SCSI,alternativa 2,tipo)
4.	Salida de Audio	Terminal 3.5 jack nivel de salida ajustable

Dimensiones y Peso

1.	Dimensiones	153 x 85 x 330mm (W/H/D)
2.	Peso	4.5 Kg

Condiciones Ambientales

1.	Temperatura de Operación	5°C ~ 45°C
2.	Humedad de Operación	30% ~ 80%
3.	Temperatura sin funcionar	-30°C ~ 65°C
4.	Humedad sin funcionar	15% ~ 80%
5.	Altitud de Operación	Horizontal ± 5°

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

1) INTRODUCTION TO AMOS
Software Manual
Alpha Microsystems
1982.

2) AMOS
User Guide
Alpha Microsystems
1986.

3) ALPHA VUE
User's Manual
Alpha Microsystems
1984.

4) ALPHA VUE/TXTFMT
Training Guide
Alpha Microsystems
1982.

5) ALPHA WRITE
Reference Guide
Alpha Microsystems
1987.

6) AMOS/L ASSEMBLY LANGUAGE
Porgrammer's Manual
Alpha Microsystems
1982.

7) ALPHA FIX/L
User's Manual
Alpha Microsystems
1982.

8) AM-100/L
Instruction Set
Alpha Microsystems
1982.

9) AMOS /L
Monitor Calls
Alpha Microsystems
1985.

10) ALPHA BASIC
User's Manual
Alpha Microsystems
1986.

11) AMOS /L ALPHA BASIC
Xcall Subroutines
Alpha Microsystems
1986.

12) ALPHA C
User's Manual
Alpha Microsystems
1985.

13) CD-ROM
Owner's Manual
XM-2000A
Toshiba Corporation
1986.

14) CD-ROM DRIVE
Owner's Manual
XM-2100A
Toshiba Corporatio
1987.

15) CD ROM
Optical Publishing
Microsoft Press
1987.