ESCUELA UNIVERSITARIA POLITECNICA

DE

LAS PALMAS

TITULO:

GESTION DE FICHEROS. APLICACION A LA CONEXION DE UN SISTEMA DE DESARROLLO A UN ORDENADOR .

AUTOR: Carlos Ley Bosch

TUTOR: Sebastián Suárez Gil

Octubre 1983

OBJETO DEL TRABAJO

El objeto de este trabajo es realizar un estudio de la informacion estructurada en forma de ficheros y de su tratamiento.En este estudio podemos distinguir dos partes bien diferenciadas:

-Una parte teórica, que consiste en el análisis de los ficheros como estructuras de datos y de los sistemas encargados de su gestión.

-Una parte practica, que trata del uso de los procesos de tratamiento del ISIS-II y de su aplicacion en un programa de comunicacion del Sistema de Desarrollo con el Ordenador Hp-3000.

La fase teorica del trabajo comienza en el capitulo primero, donde se estudiarán los conceptos basicos de los elementos de un fichero y las clases de ficheros segun sea su estructura, ademas de darse una vision general de otras formas de estructuras de datos.

En el capitulo segundo se analizaran los dos medios fisicos de mas uso en el almacenamiento de la informacion: el disco y la cinta, asícomo la forma en que se encuentra un fichero dispuesto físicamente en estos soportes.

En el capitulo tercero se tratara la manera en que se desarrollan las gestiones dentro de un sistema de ficheros, de los distintos niveles que hay en estos procesos y de como se relacionan. Como una aplicacion de todo el estudio teorico veremos en el capitulo cuarto un ejemplo de sistema de gestion de datos, el del ordenador VAX de DIGITAL.

A partir del capitulo quintoz entramos en la fase practica del trabajo. En este capitulo se decriben los procesos disponibles dentro de sistema operativo ISIS-II para realizar operaciones con ficheros (accesibles desde un programa del usuario) y mas concretamente con los ficheros del disco .

Como ejemplo de todos estos procesos en el capitulo sexto tenemos un proggrama de aplicac ion que nos servirá de base para analizar mas tarde el programa de comunicacion con un ordenador. En los capitulos septimo y octavo entramos de lleno en el programa de comuni cacion -QUILEY- y se estudian su algoritmo, funcionamiento, y todos los procesos de gestion de ficheros que hay en él.

En este trabjo queda expuesto todo el estudio que se realizó para el tratamiento de los ficheros del disco y su tratamiento, este estudio es ampliable, quedando una puerta abierta para la posible introduccion de futuras mejoras dentro de nuestro programa .

INDICE

CAPITULO I :

FICHEROS , ESTRUCTURAS DE DATOS Y BASE DE DATOS .

1.1 ELEMENTOS DE UN FICHERO

Un fichero es una colección de datos con una estructura definida formado por registros,que constan de campos,que a su vez constan de grupos de caracteres.

DATOS: La unidad más pequeña de almacenamiento de información es el caracter.Los grupos de caracteres - con un significado - se llaman campos. Grupos de campos forman un registro lógico (como el de la Fig. 1) , el registro contiene todos los datos de interés acerca de algún elemento del fichero.

La clave de un registro es un campo de interés primordial,así,algunos fi - cheros se organizan según una clave,por ejemplo : Un listín telefónico se ordena según los apellidos (clave principal) y dentro del mismo apellido según el segundo apellido (clave secundaria).

SISTEMAS DE ALMACENAMIENTO: Los ficheros se almacenan en sistemas de almacenamiento secundario porque son más baratos que la memoria principal del ordenador y tienen mayor capacidad,y también porque los datos normalmente no van a ser necesarios siempre.

Llamamos registros físico a los grupos de caracteres transmitidos entre el ordenador y el fichero.Entre cada registro físico hay un espacio de separación y para reducir el número de éstos espacios agrupamos los registros lógicos,usando el espacio de almacenamiento mucho más eficazmente.

En cada operación de lectura/escritura,se transmite un registro físico.

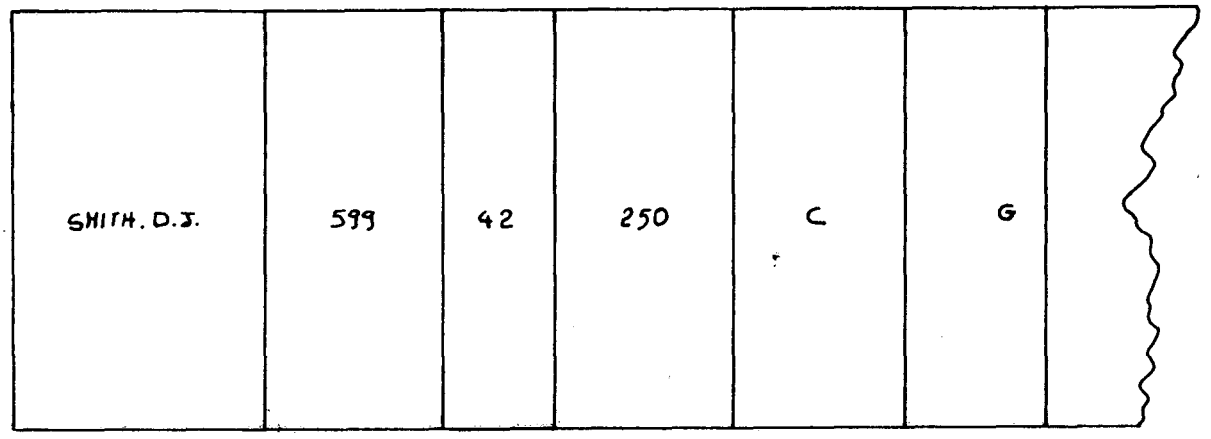
TIPOS DE REGISTROS: Podemos dividir entre:

- Registros de longitud fija: Conocemos de antemano el tamaño y el número de los campos del registro,fijamos el espacio de los datos de interés, como el ejemplo de la Fig.

- Registros de longitud variable: Este tipo de registro añade complejidad a los programas,en general,debe indicarse con un código para indicar el contenido (número de campos y tamaño) , ver Fig. 2
  El primer número del código, 4 ,da la longitud del código en caracteres, y el resto indica las variables y su tamaño.
  Como variante,podemos usar formatos standard,por ejemplo , 100,250 y 2000 caracteres,así,podemos combinar y tener varios tamaños,éste método añade complejidad al diseño y la programación,pero ahorra espacio.

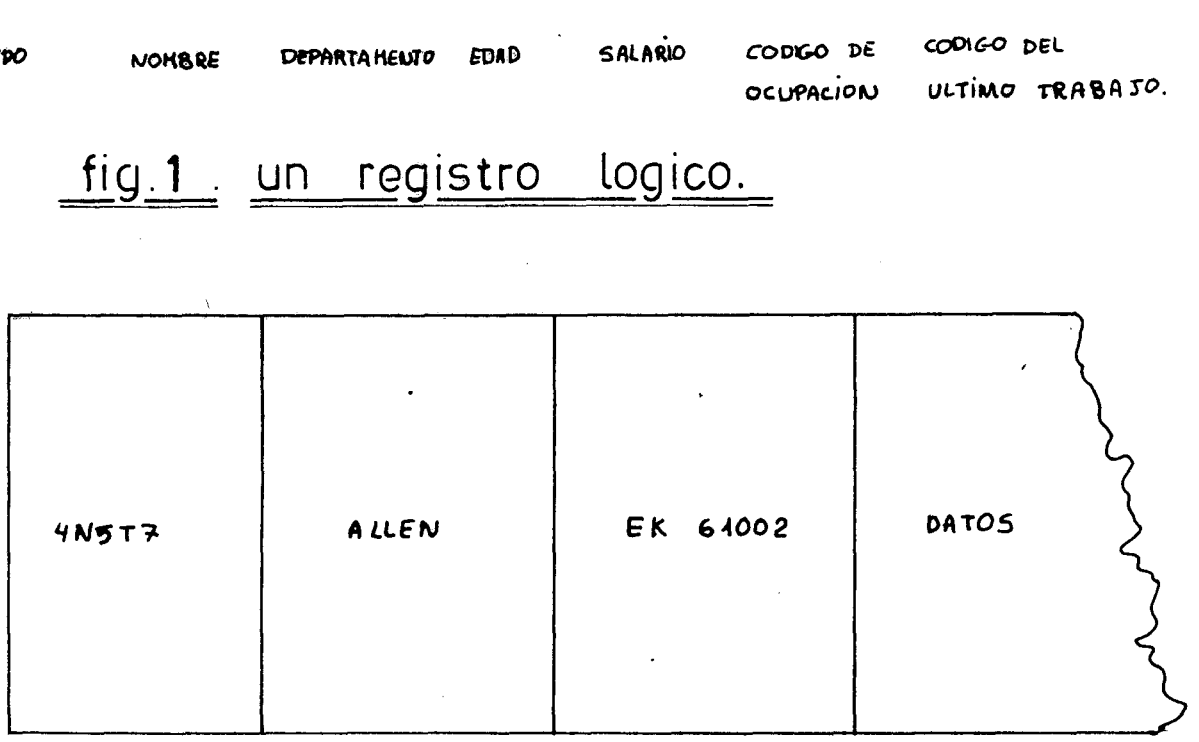
Otra alternativa a los registros fijos y variables puede usarse cuando pre-

EJEMPLO:



CAMPO | NOMBRE | DEPARTAMENTO | EDAD | SALARIO | CODIGO DE OCUPACION | CODIGO DEL ULTIMO TRABAJO.

fig.1 . un registro logico.



CODIGO | NOMBRE | REGISTRO DE COMPROBACION

LONGITUD DEL CODIGO = 4
LONGITUD DEL PRIMER NOMBRE = 5

fig.2 . codigo de registro de longitud variable

cisamos de un número variable de registros de longitud fija,por ejemplo : Una empresa dónde se precise saber los departamentos dónde ha trabajado un empleado, entonces usamos un registro encabezador (master) para los datos personales y un número variable de registros detallados (detail) según el número dedepartamentos dónde halla trabajado.

1.2 FICHEROS SECUENCIALES

Son el tipo más sencillo de fichero en el que todos los registros están en secuencia según un código.
Muchas aplicaciones de los ordenadores se basan en los ficheros secuenciales,y éstos se seguirán usando en el futuro.

MEDIO DE ALMACENAMIENTO: Los ficheros secuenciales se almacenan generalmente en cinta magnética,los datos se ordenan según un código (un listin telefónico según el orden alfabético).
En el capítulo de sistemas de memoria auxiliares haremos un estudio de las cintas magnéticas como soportes de ficheros secuenciales.

PROCESAMIENTO DE FICHEROS SECUENCIALES: El procesamiento implica el ordenamiento de los datos según algún código (numérico,alfabético) ya que los ficheros están ensecuencia.
Los fabricantes suelen suministrar como parte del sistema operativo del ordenador un " sistema de ordenamiento ",el cual indicándole los campos,tamaño de registros,y clave de ordenamiento asigna las areas de trabajo del fichero.

MODIFICACIONES EN FICHEROS SECUENCIALES: En la Fig. 3 tenemos el esquema de cómo renovar un fichero secuencial.
Como el fichero original está ordenado,las operaciones (modificaciones) de entrada deben ordenarse en el mismo orden que el fichero a procesarlo.

Hay que hacer notar que lo que se hace al renovar es crear un nuevo fichero,ya que no son posibles las insersiones en el fichero almacenado en cinta.El fichero original nos puede servir para que en caso de errores o destrucción accidental del fichero renovado,recuperemos la información y podamos volver a modificar.
En una modificación hay tres acciones posibles:

- Modificar un registro (cambiar parte de él e insertarlo en el nuevo fichero .)

- Añadir un registro (en la secuencia apropiada del nuevo fichero) .

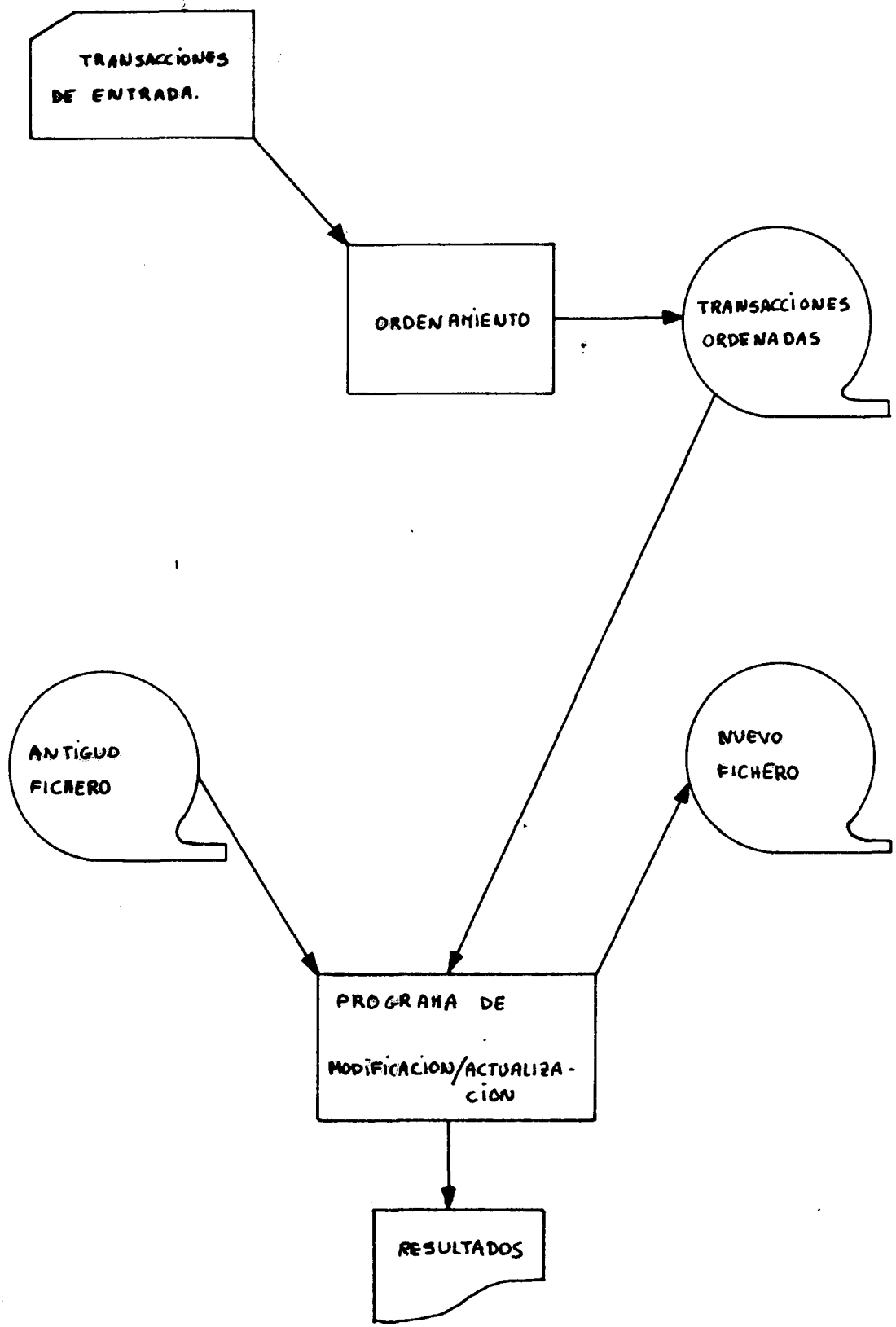


fig. 3. esquema de modificacion de un fichero secuencial

- Borrar un registro (no incorporarlo en el fichero renovado).

- Las modificaciones referentes a un registro en particular deben seguir la secuencia:
  (1) Añadir , (2) Modificar y (3) Borrar.
  Esto hace que la lógica de modificación de un fichero secuencial sea bastante complicada.
  El programa de la modificación debe ser diseñado de tal modo que durante la misma sea posible cambiar cualquier campo del registro para corregir errores.
  Las inserciones,modificaciones y supresiones deben hacerse en la misma secuencia que la del fichero que se renueva,para hacer las transacciones en el mismo orden,usamos un programa de ordenamiento.

RECUPERACION: La recuperación de la información de un fichero secuencial es una de las mayores desventajas de los ficheros secuenciales en cinta es que hay que procesar todo el fichero para acceder a la información.
Si sólo queremos acceder a unos pocos registros es necesario leer todo el fichero.
Un fichero secuencial acaba con un registro físico especial llamado marca de FIN DE FICHERO.
Si aumentamos el número de registros de un fichero secuencial,éste debe ser pasado a una nueva posición mayor en la memoria auxiliar.
Si usamos discos,podemos evitar ésto mediante el uso de registros encadenados,se coloca en cada registro la dirección del siguiente registro contíguo,de éste modo,el espacio para un fichero secuencial encadenado puede ser asignado en cualquier parte del disco.La marca de FIN DE FICHERO vendrá entonces en el último registro físico,en vez de la dirección del registro siguiente.
Para un acceso secuencial más rápido las direcciones de encadenamiento pueden colocarse en una lista aparte llamada lista de E/S , la lista estará a su vez almacenada en registros físicos encadenados.

1.3 FICHEROS DE ACCESO DIRECTO

Para solucionar algunos de los problemas de los ficheros secuenciales y proveer una recuperación de la información más rápida se usan los ficheros de acceso directo, estructuras de almacenamiento más complejas,pero que dan una mayor flexibilidad de manejo de la información.
Cada registro lógico de un fichero puede asignarse a un registro físico dado,si la dirección del registro físico puede obtenerse a partir de la clave del registro lógico,se puede acceder directamente sin necesidad de

ninguna exploración.
Tales ficheros se llaman directamente indexados, éste se usa en discos dónde los sectores pueden direccionarse directamente.
La correspondencia entre las claves y las direcciones del disco puede obtenerse manteniendo en memoria principal una lista de claves y sus direccio - nes correspondientes, si los registros lógicos están agrupados se necesita una búsqueda secuencial para localizar uno determinado.
La lista de claves y sus direcciones son los elementos básicos de un tipo de fichero llamado indexado, en el que la lista se almacena en memoria auxiliar. A ésta lista se le llama índice (Fig. 4).
Así, con un valor de la clave, la dirección del registro físico puede encontrarse buscando en el índice, luego, encontramos el registro lógico dado explorando el registro físico secuencialmente.
Vemos que para cada registro físico examinado necesitamos un acceso, cada vez, a la memoria auxiliar, para evitar largas busquedas secuenciales del índice usamos varios niveles de indexación. Cada índice de nivel superior contiene una lista de los últimos valores de la clave presentes en cada registro físico del siguiente índice de nivel inmediatamente inferior (Fig. 5).
Los registros físicos pueden almacenarse secuencialmente en direcciones consecutivas, en cuyo caso el fichero se puede acceder al fichero secuencialmente o por indexación. Tales ficheros se conocen como secuencial - indexados.

PROCESAMIENTO DE FICHEROS DE ACCESO DIRECTO: No hay razón por la que un fichero de acceso directo no pueda procesarse secuencialmente de la misma forma descrita en el apartado de ficheros secuenciales.
Pero, cuando procesamos los ficheros directamente ¿ cómo localizamos el registro requerido ? Si queremos acceder a un registro determinado el software de gestión del fichero nos lo proporcionará. Sin embargo, debemos asociar el número de registro lógico con la información deseada. Por ejemplo, en un inventario ¿ Cómo sabemos dónde está la información de la pieza 1432 ?, ¿ Qué registro lógico contiene los datos acerca de la pieza 1432 ? Para ello se usa un directorio, y relaciona nuestra referencia de interés (pieza 1432) con la dirección de su registro lógico del fichero, a ésto se le llama transformación referencia - dirección y hay tres tipos de directorios que hacen ésta transformación :

a) Directo : Raramente usado, aquí es la propia referencia la dirección, por ejemplo, la pieza 10 se almacena en el registro 10.

b) Diccionario : Es el más usado. Un diccionario - una tabla en la memoria - relaciona las referencias y su localización, por ejemplo:

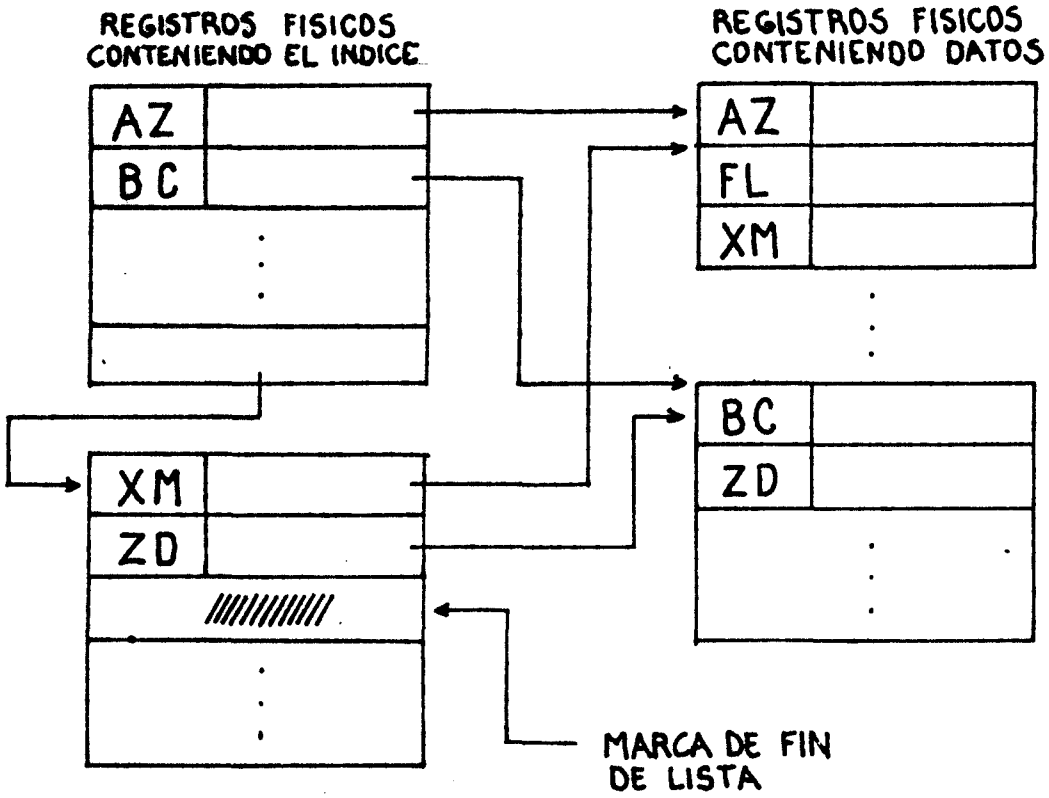


fig.4 un fichero indexado

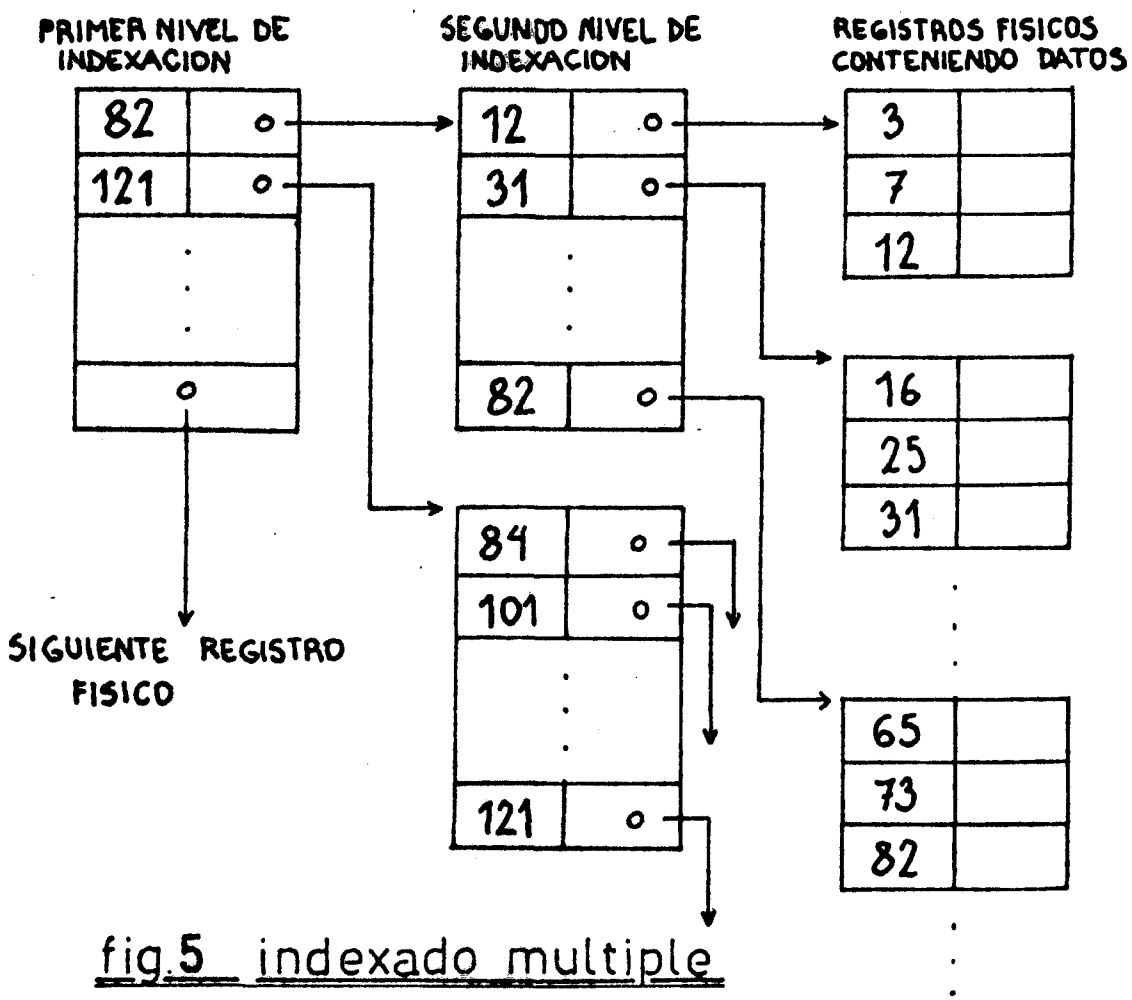


fig.5 indexado multiple

(Fig. 6).

Miramos en el diccionario de la memoria principal, buscando la referencia, la entrada del diccionario nos dice en que registro se encuentra. En el caso de un gran fichero, el diccionario es tan grande que se almacena en el disco, y se llevan partes de él a la memoria principal para la búsqueda.

En la búsqueda BINARIA dividimos el diccionario por la mitad y comparamos la entrada de valor medio con la referencia a encontrar, si la referencia está en la mitad superior de la tabla, dividimos ésta de nuevo en dos. A cada comporación, se reduce el número de entradas del diccionario en la mitad. En la tercera comparación sólo nos queda un octavo del diccionario, cuando nos quedan unas pocas entradas, ya es posible buscar secuencialmente. El diccionario está almacenado en disco los valores con los que se compara la referencia a buscar (o sea las posiciones 1/2, 1/4, 1/8) se almacenan en memoria principal, se compara la referencia con esos números y se trae a memoria aquel 1/8 del diccionario dónde se encuentre la referencia para hacer una búsqueda secuencial.

c) Desmenuzamiento (hashing) : Es el último tipo de técnica de transformación de la referencia a dirección.

Aquí ganamos velocidad de acceso a expensas del espacio de almacenamiento. Se trata de realizar cálculos con la referencia y usar los resultados de éstos cálculos como dirección. Puede suceder lo que se llama " colisión ": Dos referencias distintas, que en éste caso se llamarían " sinónimos ", den como resultado la misma dirección, en éste caso podemos hacer un nuevo cálculo de la dirección o bien buscar en siquiente registro libre del fichero y colocar los datos allí. Los " sinónimos " crean dificultades a la hora de modificar éste tipo de ficheros, supongamos que dos referencias se direccionan al registro lógico 2365 y si el registro 2366 está vacío, la segunda de las referencias se almacenará en el registro 2366, si, más tarde borramos el primer registro (en 2365), sucederá que perderemos el registro 2366. Esto sucede porque en realidad ambos tienen la misma dirección después de la operación de direccionamiento, la " colisión " nos obligó a colocar el segundo registro en una posición contigua, 2366. Entonces, al examinar la posición 2365 del fichero y encontrarla vacía suponemos que el segundo registro no está en el fichero.

En otras palabras, borrar físicamente el primer registro implica que destruímos la vía hacia el segundo.

Para evitar ésto, usamos un indicador de borrado en cada registro para indi-

car cuando se borra,y periodicamente se reestructura el fichero físicamente y los registros borrados se eliminan y se asignan nuevas posiciones a los registros en uso.
Así,en nuestro caso el registro 2366 pasará a la 2365.

En cuanto a los cálculos,el más usual es dividir la referencia por el mayor número primo más pequeño que el tamaño del fichero en registros,y usar el resto como dirección,el objetivo es conseguir siempre un número mínimo de colisiones.

OTROS ACCESOS MAS COMPLEJOS: Supongamos un inventario y queremos saber que partes pertenecen a cada montaje.Tal como vemos en la Fig. 7 ,necesitaremos un modelo de fichero que nos responda a preguntas como qué partes pertenecen al montaje 103.Para ello podríamos ir leyendo todos los registros uno por uno,pero puede ocurrir que entre dos registros que contengan partes del montaje 103 hayan cientos de otros ,luego no es un sistema rápido ni eficiente. Entonces se usa un puntero que es una parte de los datos cuyo valor es la dirección de otro registro,el siguiente dónde haya una parte del 103,así el fichero quedaría como en la Fig. 8
El puntero del registro 1 apuntará al registro 4 ,que es el siguiente dónde hay partes del montaje 103,y así sucesivamente.Este tipo de fichero se llama encadenado.¿ Y que hacemos para encontrar el registro con la primera pieza del montaje 103 ? Esto se resuelve usando un directorio como el de la tabla 9 , nos indica el número del registro conteniendo la primera pieza,luego,para seguir vamos continuando por la cadena de punteros.
También podemos sacar los punteros en el directorio,que entonces se llama directorio invertido . (Fig. 10)

MODIFICACION: Modificar un fichero de acceso directo puede hacerse directa o secuencialmente.
Si modificamos directamente,sólo los registros que se cambian necesitan modificarse,no es necesario procesar todo el fichero.El inconveniente de ésto es que es fácil perder un control de verificación ,los registros se cambian y no hay copia como había en los ficheros secuenciales.
¿ Qué le sucede a los punteros cuando añadimos,modificamos o borramos registros en un fichero de acceso directo ?
Si,en la tabla la pieza número 3218 pasa a ser del montaje 103 al 607.
¿ Basta con cambiar el campo del registro 4 del montaje y poner 607?
Si no hubieran punteros,la respuesta sería si , pero esa modificación destruiría nuestra secuencia de punteros,para evitar ésto,podemos hacer:

REGISTRO NO.	PIEZA NO.	MONTAJE	EXISTENCIAS	VENDEDOR
1	4326	103	27	ACME
2	6742	607	51	JOHNSON
3	8137	12	100	DAWES
4	3218	103	13	FRAZIER
5	3762	607	43	ARMOR

fig.7. ejemplo de fichero

REGISTRO	PIEZA	MONTAJE	EXISTENCIAS	VENDEDOR	PUNTERO
1	4326	103	27	ACME	4
2	6742	607	51	JOHNSDN	5
3	8137	12	100	DAWES	13
4	3218	103	13	FRAZIER	42
5	3762	607	43	ARMOR	103

fig.8. ejemplo de fichero

MONTAJE	REGISTRO
12	3
25	212
103	1
104	62
607	2

fig.9. directorio de montajes

MONTAJE	REGISTRO
12	3, 13 ...
25	212 ...
103	1, 4, 42 ...
104	62 ...
607	2, 5, 106 ...

fig.10. dir. invertido de montajes

a) Realizar un programa que trace la cadena de punteros de registros del montaje 103,y encuentre el que apunta al registro 4,entonces,se cambiaría el puntero del registro 1 para que apunte al 42 (que es el siguiente después del 4) .

b) Diseñar un fichero con punteros "retrovisores",por ejemplo un puntero en el registro 4 que apunte al 1 (el inmediatamente anterior),así,con los dos tipos de punteros será solamente necesario acceder a los tres registros envueltos en el cambio (anterior - modificado - posterior).

c) Crear un indicador de borrado y dejar el registro en el fichero.En éste caso, colocamos un indicador de borrado en el número antiguo de la pieza (3218 en el registro 4) y añadimos un nuevo registro para la parte 3218 perteneciente al montaje 607 . Como el fichero se reestructura periódicamente,se eliminan los registros con indicadores de borrado y se colocan nuevos punteros y directorios.

Estas tres posibilidades nos modifican el registro 4 sin variar la cadena de punteros del montaje 103,el último paso será modificar el registro 4 y meterlo en la cadena del montaje 607 ,ésto lo hacemos haciendo que el directorio apunte ahora al registro 4 y el puntero de éste último apunte a la antigua entrada del directorio del montaje 607 , o sea , el registro 2.

1.4 CONTROL DE ERRORES: En cualquier operación con ficheros,debemos tener algún tipo de recuperación o control de la información.En un fichero secuencial, como hemos visto,se mantiene el antiguo fichero no modificado.En un fichero de acceso directo tenemos que depurar los ficheros periodicamente si no se modifican secuencialmente los registros.

El contro de todos los procesos es necesario para mantener la integridad del fichero.Se debe comprobar si en una transferenci los campos que precisan de datos numéricos reciben datos numéricos.

Para campos que sean especialmente cruciales en el fichero,se deben verificar todos los cambios.Por ejemplo,un programa que modifique un fichero debe llevar un registro resúmen al final del fichero con varios totales,si el fichero es un inventario se debería tener el total del número de unidades.

Durante la modificación del fichero el programa sumará todas las partes, teniendo el control de las añadidas y las que ya estaban.

Al final de la modificación el registro resúmen se examinará para ver si el total antiguo más las partes nuevas es igual al nuevo total.

1.5 ESTRUCTURAS DE DATOS: Representan las relaciones físicas y lógicas de los

registros de un fichero.
Una estructura de datos física es simplemente la forma en que se encuentran los datos en la unidad de almacenamiento :

- En un fichero secuencial: Los registros están colocados en secuencia.
- En uno de acceso directo: La estructura física incluye algunos registros directorio al principio seguidos de registros de registros con los datos.

- De más interés son las estructuras de datos lógicas,podemos crear varios tipos según el tipo de aplicación del proceso.De hecho,ya hemos visto algunos tipos de estructuras lógicas,por ejemplo,hemos dicho cómo van los punteros en un fichero,una visión desde el punto de vista de estructura lógica de un fichero podría usar flechas para indicar las conexiones entre los diversos campos.
  Hemos visto una estructura de " lista ",que la podemos esquematizar para ver las relaciones lógicas entre los registros y queda como la Fig. 11
  Una lista puede fácilmente extenderse y llegar a ser un anillo, como el de la figura 12 . En un anillo,el último registro de la lista apunta al primero,que contiene un símbolo para indicar que es el primero. Podemos seguir el anillo para encontrar cualquier registro,por ejemplo, el registro precedente,el siguiente,o el primero del anillo.

  En general tenemos tres tipos de estructuras de datos:

  a) Estructura jerarquizada: También llamada estructura de árbol.Fig. 13
  El árbol se compone de una jerarquía de nudos ,el nudo superior se llama raíz.Cada nudo (menos la raíz) está relacionado con un nudo de nivel superior llamado " padre ",ninguno puede tener más de un padre,pero si puede tener más de un nudo inferior,llamado " niño ".
  Un fichero jerarquizado es el que tiene una relación estructurada en árbol entre sus registros,ésta estructura es muy conveniente porque muchos datos tienden por naturaleza a estar jerarquizados.

  b) Estructura en red: Sucede cuando un " niño " tiene más de un " padre " en la estructura de relación de datos.Así,un elemento de ésta estructura puede unirse con cualquier otro.La estructura física para el soporte para éstas estructuras es muy complicada,como

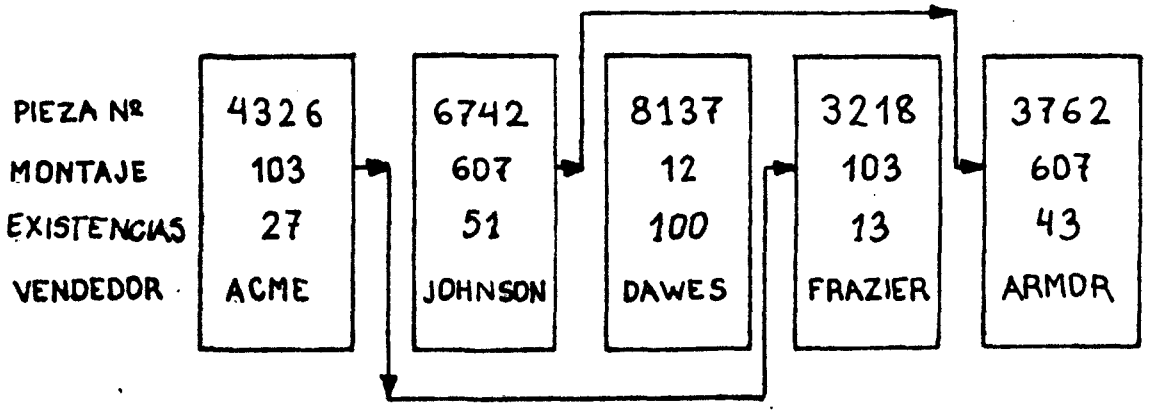


fig.11. estructura de fichero de la tabla

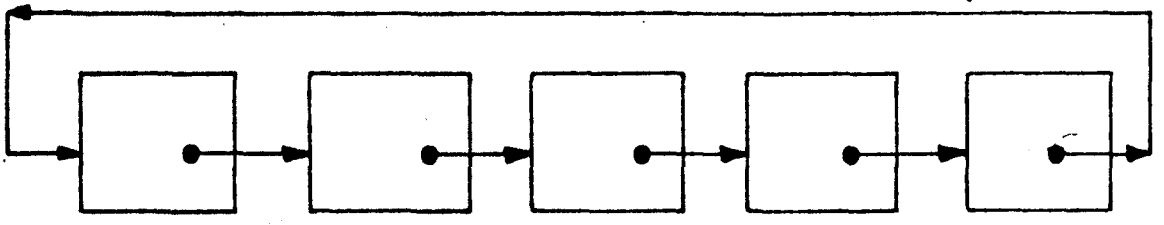


fig.12. estructura en anillo.

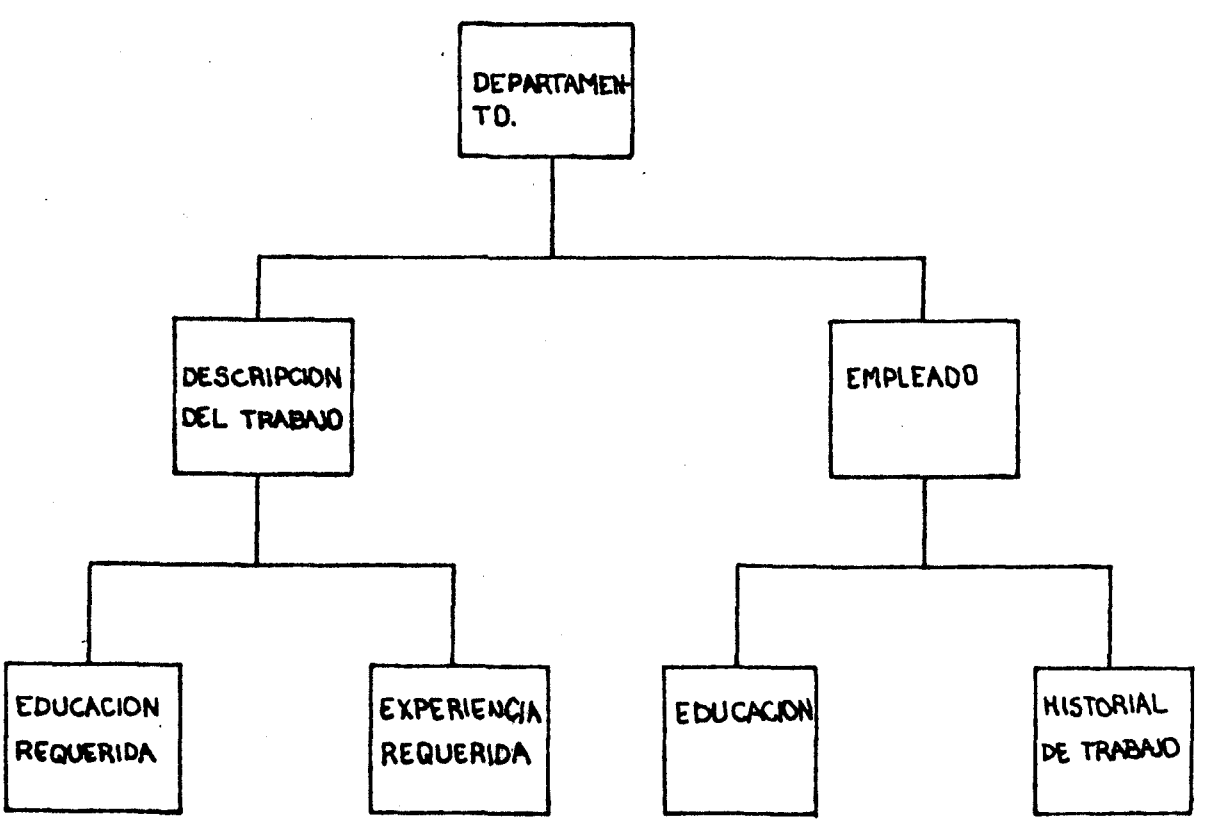


fig.13. estructura jerarquizada o en arbol.

podemos ver estudiando la figura 14 .

c) Estructura " relacional " : Su idea básica es muy simple, los datos se organizan en tablas de dos dimensiones como la de la Fig. 15 . Tales tablas son fáciles de entender y de desarrollar por el usuario.

Una virtud de éste tipo de estructura es que puede descrivirse matemáticamente, cosa muy complicada en los otros tipos.

El nombre se diriva del hecho de que cada tabla representa una relación.

Como diferentes usuarios ven diferentes grupos de datos y relaciones entre ellos, es necesario extraer partes de las columnas de la tabla para algunos usuarios y unir tablas para otros para hacer tablas mayores. Esta capacidad de manipular las relaciones provee una flexibilidad que no esdisponible en las estructuras jerarquizada y en red.

1.6 SISTEMAS DE TRATAMIENTO DE BASE DE DATOS

El objetivo de un sistema de tratamiento de base de datos es facilitar la creación de estructuras de datos y relevar al programador de los problemas de tratar con ficheros complicados. Además, se consideran los datos como un recurso en la organización del sistema y como algo que debe ser manejado con cuidado.

Los primeros sistemas de tratamiento de ficheros operaban en ficheros secuenciales. Los usuarios describian los registros, y se usaba un lenguaje para expresar las complicadas relaciones entre los campos, muchos de éstos sistemas se han ampliado para incluir capacidad de modificación.

Cuando se hicieron comunes los ficheros de acceso directo, varios vendedores ofrecieron sistemas de acceso a ficheros para uso en los programas escritos en lenguajes de ordenador, como el COBOL . Uno de esos metodos es el " metodo de acceso secuencial indexado " (o ISAM) . Este software permite al programador de COBOL desarrollar un programa para acceder y modificar la información de un fichero de acceso directo con una sola clave de acceso, sin construir directorio.

El ISAM mantiene un directorio para cada registro, el programa de aplicación proporciona la clave y el sistema accede al registro.

El software mantiene areas de desbordamiento y punteros para tener el fiche ro ordenado secuencialmente. Así, las modificaciones se pueden hacer secuen-

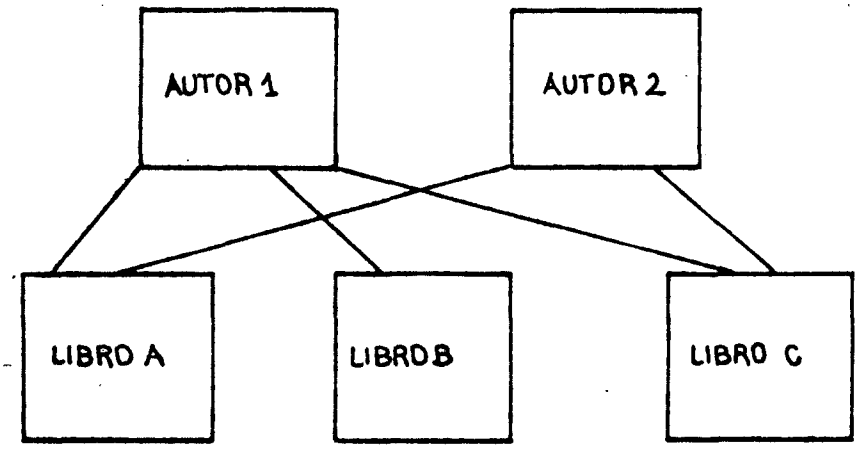


fig.14. estructura en red.

NOMBRE	DIRECCION	APARTADO	CIUDAD	DEPARTAMENTO (Nº)
SMITH	16 MAIN	16	NEW YORK	302
JONES	37 SPENCER	01	CHICAGO	161
MORRIS	19 OLD WAY	24	NEW YORK	302
ABLE	86 FULTON	06	DENVER	927
CHARLES	19 HUNTER	26	CHICAGO	161

NOMBRE	PROFESION	INGRESOS
JOHNSON	COMERCIANTE	15.000
MARTIN	PROGRAMADOR	14.000
JONES	ANALISTA	18.000
CARSON	MANAGER	17.000
SMITH	ANALISTA	19.000

SOCIEDAD:

NOMBRE	DIRECCION	APARTADO	PROFESION	INGRESOS
JONES	37 SPENCER	01	ANALISTA	18.000

PROYECTO:

CIUDAD	DEPARTAMENTO
NEW YORK	302
CHICAGO	161
DENVER	927

fig.15. estructura relacionada.

cialmente,y el acceso puede ser directo.Cuando un fichero se varía en orden y se llenan las zonas de desbordamiento,hay que reestructurarlo periodicamente.

Metodos de acceso como éste,y los sistemas de tratamiento de ficheros han evolucionado hacia los sistemas complejos de base de datos.

1.7 SISTEMAS COMPLETOS DE BASE DE DATOS: Un sistema de tratamiento de base de datos es una parte de software que permite a una organización desarrollar aplicaciones con base de datos.

Un sistema completo de base de datos separa entre los datos y los programas que acceden a éstos.

Si se usa un sistema de datos es necesario que las mismas unidades reciban siempre la misma nomenclatura,no es posible llamar a un fichero SALARIO en un programa y NOMINA en otro.

Para algunas organizaciones la necesidad de desarrollar un diccionario común de las unidades de datos es una de los mayores costos de adquirir el sistema de data-base.

El administrador de data-base ,o los programadores,deben usar un lenguaje especial de definición de datos para definir las estructuras de datos y ficheros en el sistema.

El lenguaje debe especificar como se agrupan los datos,especificar las subdivisiones de los registros según las estructuras lógicas, y debe ser posible dar nombres a las relaciones entre agrupaciones de datos.

Con las definiciones de datos y la data-base,el sistema de data-base construye todos los punteros,encadenamientos,y directorios automaticamente.

El programa de aplicación pide los datos que necesita, el sistema de data-base examina la petición de datos y determina donde están localizados los registros de interés,y manda el registro o el campo pedido al programa.

Con uno de éstos sistemas,es posible diseñar estructuras de fichero mucho más fácilmente y desarrollar una data-base que se pueda usar por diferentes programas de aplicación.

Estos sistemas evitan la redundancia de datos,los mismos datos no son mantenidos por distintos sistemas, cada uno con ficheros diferentes.

Muchas organizaciones han obtenido resultados impresionantes usando sistemas de data-base ,sim embargo,es necesario estudiar y evaluar los sitemas con cuidado,si no necesitamos el más complejo,vayamos a otros más sencillos.

Las tendencias en el futuro de los sistemas de data-base es de su uso para ahorrar tiempo al programador,al analista y en la implementación.

En general hay pocos standards en base de datos y bastantes diferencias entre ellos.

Ejemplo:

Los diferentes sistemas de data-base ofrecen una variedad de estructuras de fichero,de modo que el mismo problema puede resolverse con la estructura de cada sistema.

Un sistema de data-base tiene dos tipos de fichero uno maestro y uno variable.Al maestro se accese mediante alguna clave y se encadena a el fichero variable.Dentro del registro variable las asociaciones entre entradas están encadenadas por los punteros.

La figura 16 muestra ejemplos de distintas estructuras lógicas de fichero.En la parte superior vemos un fichero maestro de las piezas de un inventario y sus cantidades en stock.Este fichero maestro está unido a un fichero variable que contiene un control de las salidas y entradas.

Un fichero maestro puede tener varios ficheros variables,en el ejemplo de la Fig. 16 ,podemos añadir otro fichero variable para controlar las recepciones de cada parte.También es posible tener dos ficheros maestros que accedan a un fichero variable, conocen la parte inferior de la figura 16 ,aquí el segundo fichero maestro,una lista de proyectos,está unido a las transacciones que tuvieron lugar para un proyecto particular.

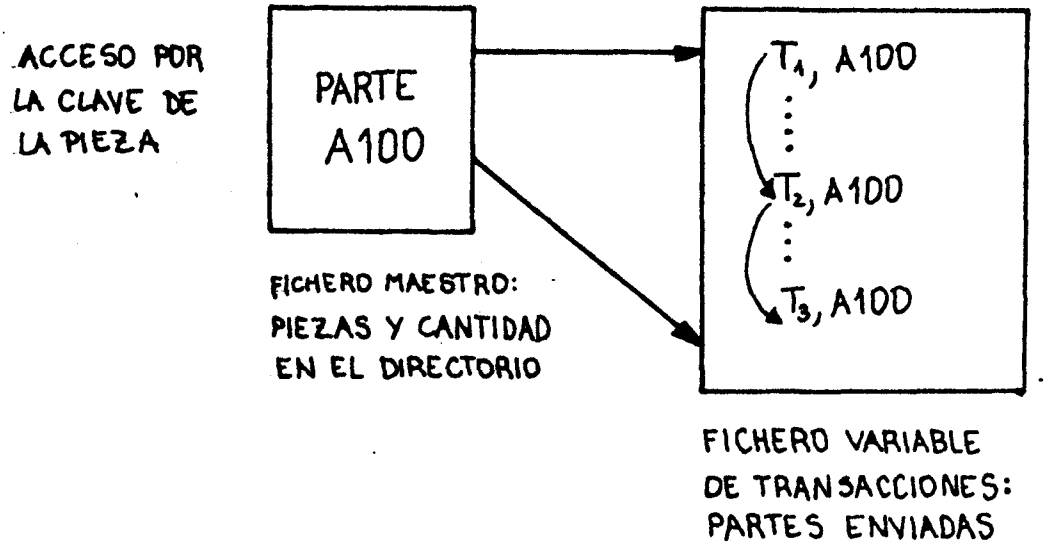
Hay,entonces, dos tipos de uniones con el fichero variable:una para las transacciones para la misma pieza y otra para las transacciones con el mismo proyecto.Esta misma estructura lógica podría representarse en sistemas de data-base usando otras formas como la de árbol jerarquizado.

1.8 CONSIDEREACIONES EN EL DISEÑO DE FICHEROS

Hasta ahora ,hemos visto el aspecto de estructura lógica de ficheros y sistemas de soporte de ficheros,pero no hemos descrito cómo el diseñador de sistemas decide que clase de estructura de fichero usar.

En ésta sección,veremos algunas líneas maestras del diseño.La mayoría de las estructuras de ficheros están en función de la aplicación,veamos algunos de los aspectos primordiales en el diseño.

a) Estructura de los registros: Es el primer aspecto que concerne con los datos a almacenar en los ficheros.En el diseño de un sistema de información se deben considerar los requerimientos para almacenar y acceder a la información.La información que tiene



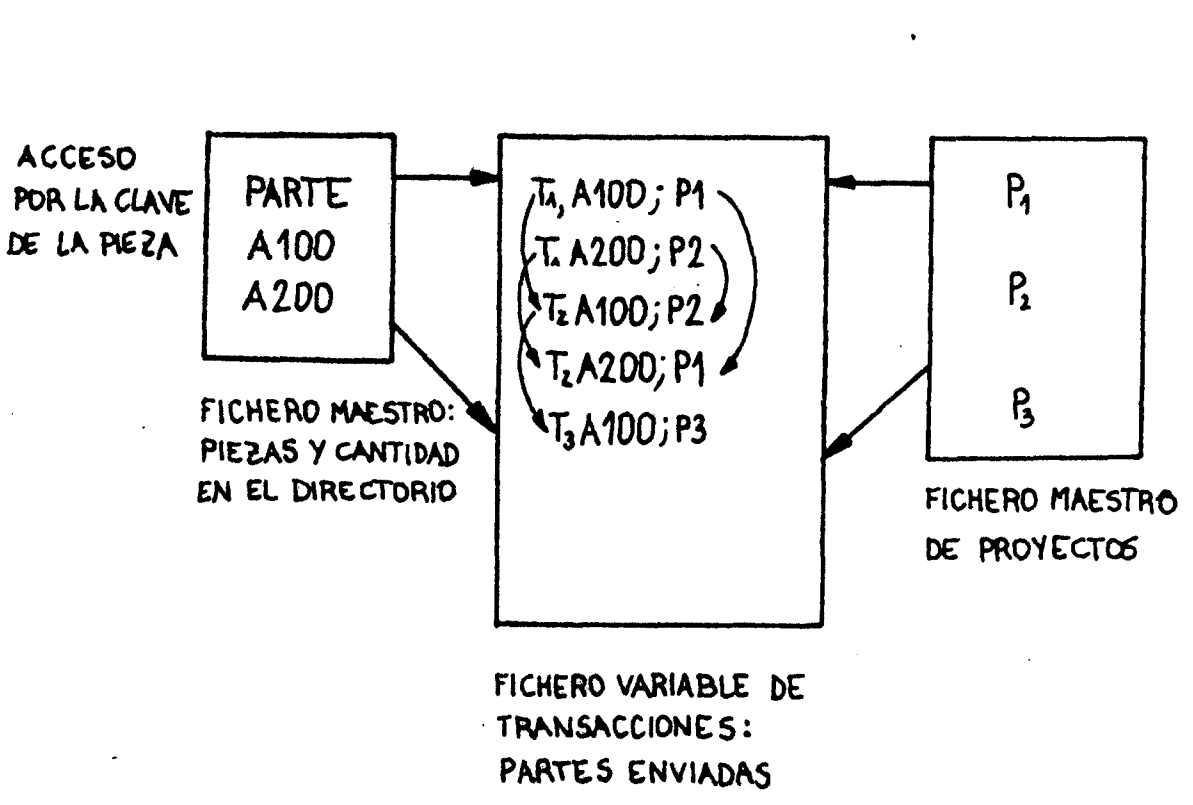


fig.16 .ejemplo de base de datos en red

interrelación se almacena en registros,y varios tipos de registros pueden haber en un fichero físico.Debe definirse el tipo de registro,así como las claves para acceder a los campos.
La información de un fichero generalmente se modifica de dos maneras: Una, mediante una rutina de transacción que cambia los campos de un fichero,por ejemplo: el recibir una nueva referencia cambia el campo de cantidad de unidades de un fichero de inventario.
Otros campos delfichero se cambian mucho menos frecuentemente,por ejemplo: la de los vendedores de artículos de almanen.
Este segundo tipo de cambio podríamos llamarlo mantenimiento de un fichero.
Habiendo definido el contenido básico de la información de los ficheros, con la agrupación de los registros lógicos,el diseñador estudia el problema de definir el formato de registro.Los registros de longitud fija son los de más fácil y simple uso desde el punto de vista de programación y procesamiento.Si hay una cantidad variable de información de longitud fija,se usan los registros maestros (encabezador) y variable que hemos visto.Sin embargo,si la longitud del registro varía,son necesarios registros de longitud variable.

b) Respuesta en función del costo: Una vez definido el contenido y formato del fichero,el analista examina los requerimientos de naturaleza,volumén y tiempo de respuesta de la información para acceso y modificación del fichero.Debemos evaluar los requerimientos del tiempo de respuesta en contrapartida con el costo de:
- Crear la base de datos.
- Almacenar los datos.
- Acceso a los datos.
- Modificación de datos.

Parte de la investigación de diseño de ficheros ha dado como resultado el desarrollo de varios modelos que ayudan al diseñador a preveer los costos de los distintos sistemas de base de datos.En el futuro,el diseñador será capaz de describir las unidades de soporte físico de ficheros y la estructura de la data-base.
Mediante el uso de un ordenador con tiempo compartido será posible comparar diferentes diseños de fichero.

CAPITULO II :

SISTEMAS DE MEMORIA AUXILIAR Y FICHEROS FISICOS .

2.1 SISTEMAS DE MEMORIA AUXILIARES

La memoria de un ordenador debería ser tan grande y rápida como fuera posible. La capacidad de almacenamiento de una memoria nos dará el número de bits que puede almacenar, y la velocidad viene en función del tiempo de acceso, que es el tiempo medio requerido para acceder o almacenar en una célula de memoria dada. Sin embargo debe haber un compromiso, pues el costo por bit de almacenamiento aumenta según disminuye el tiempo de acceso. Como consecuencia, se establece una gerarquia en los sistemas de memoria: (Fig 17)

- Una cantidad limitada de memoria muy cara.
- Una mayor cantidad de memoria más barata.
- Gran cantidad de memoria de muy bajo coste.

El nivel más caro de sistemas de memoria lo componen los registros del procesador, estas células se usan para almacenar los datos manejados por el procesador durante la ejecución de un programa.

El segundo nivel es la memoria principal del ordenador.

Los niveles inferiores representan la memoria auxiliar; el cuarto nivel de la jerarquia consiste en las unidades de cinta, que son sistemas de acceso secuencial. Los datos se almacenan en cinta magnética frente a una cabeza de escritura que magnetiza pequeñas areas de la superficie de la cinta; cada porción se magnetiza en una dirección, según almacenemos un 1 ó 0.

La cinta contiene entonces una secuencia de caracteres (8 bits), y se lee pasandose por una cabeza de lectura que detecta los datos almacenados (Fig 18).

Los datos se almacenan en una secuencia de caracteres, llamada bloque. En la lectura se accede a los datos leyendo bloques completos. A causa de las caracteristicas del mecanismo de la cinta, los bloques se separan con espacios. Para aprovechar mejor la cinta, se agrupan los datos en grandes bloques.

Las unidades de cintas permiten cinco operaciones:

- Rewind: Rebobinado de la cinta para que el primer bloque esté frente a las cabezas de lectura/escritura.
- Read: lectura del bloque que esta frente a las cabezas.
- Write: Escritura de un bloque en la posición dada de la cabeza.
- Backspace: Rebobinar la cinta y dejar las cabezas colocadas en el bloque precedente.

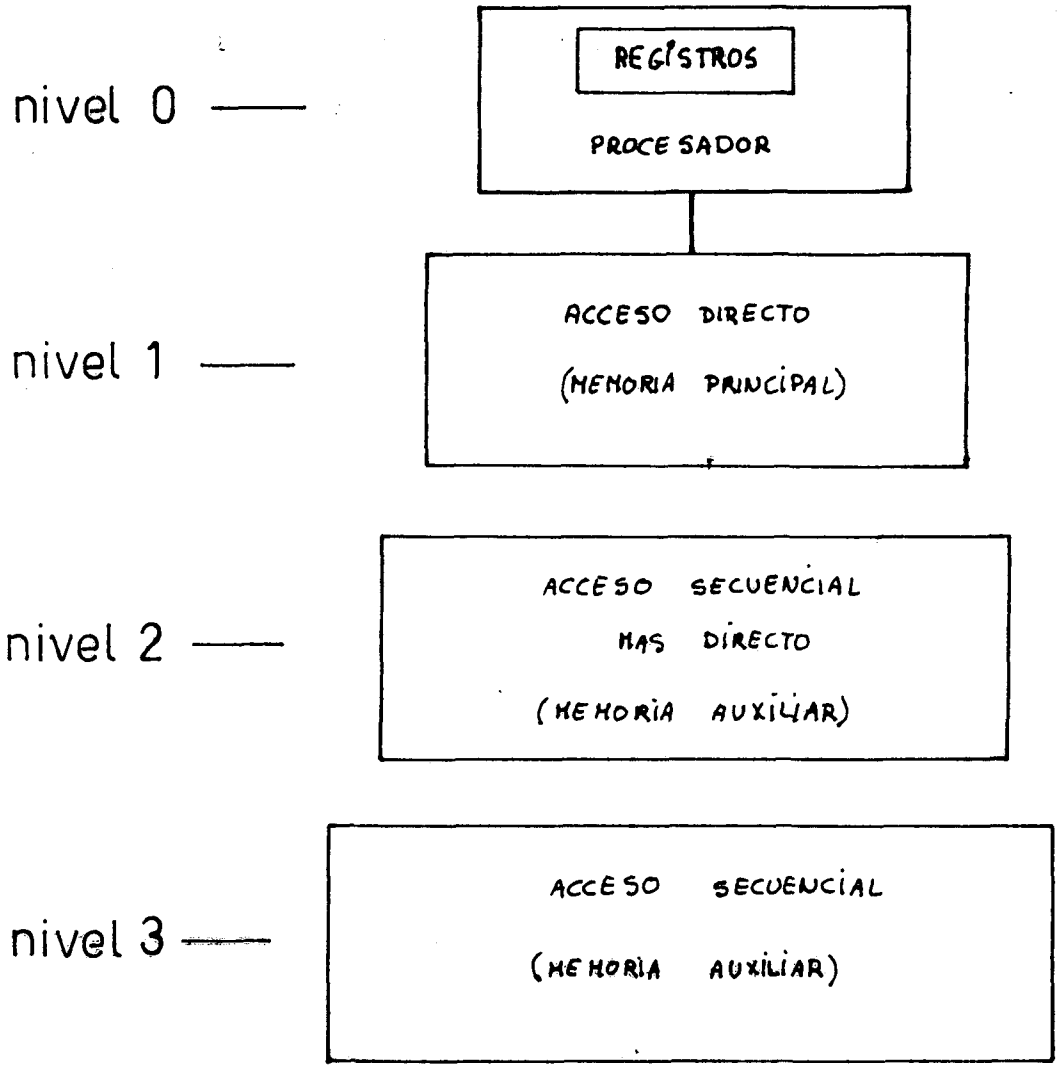


fig.17 . jerarquia de los sistemas de memoria

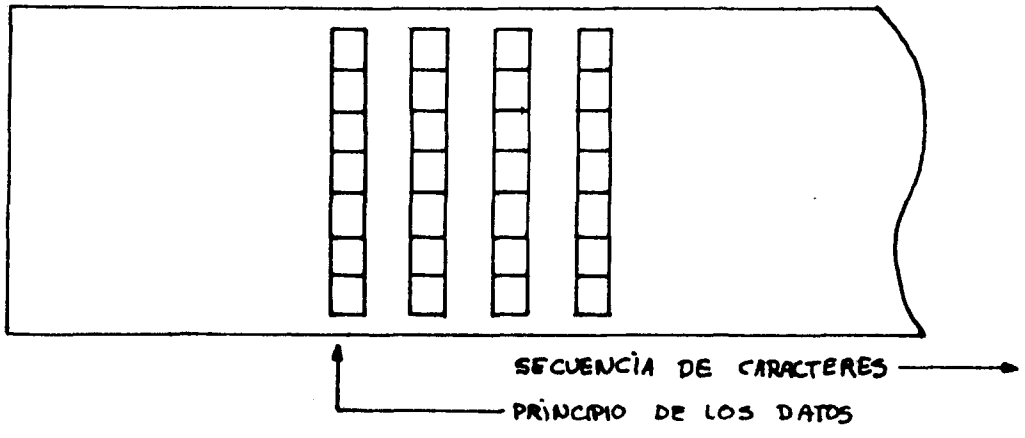


fig.18. memoria de cinta magnética

- End file: Escritura de un bloque especial llamado marca de FIN DE FICHERO.
- La densidad de almacenamiento de la cinta es el número de caracteres que se graban por pulgada (o también bits por pulgada).

Una secuencia de bloques determinada por una marca FIN DE FICHERO constituye un fichero físico. A causa de las inexactitudes de la mecanica de las unidades de cinta, no es posible escribir un bloque en la cinta a la mitad de un fichero, incluso si los bloques son de igual longitud, el bloque modificado no coincidirá probablemente con el antiguo. El bloque es la unidad de transferencia de información hacia y desde los sistemas de memoria auxiliares. En las cintas, el término bloque es equivalente a registro físico, que es la unidad de datos almacenados direccionable en los sistemas de memoria auxiliar. Los registros físicos proporcionan el medio de almacenamiento para los bloques. Un bloque puede ocupar uno o más registros físicos.

El tercer nivel en la jerarquia de memorias es el disco magnético, que combina los modos de acceso secuencial y directo. Uno de los modos más comunes es el disco de cabeza móvil (Fig.19), contiene un juego de platos rotatorios cubiertos de material magnético. Los datos se escriben y leen mediante un juego de cabezas de lectura/escritura unidos a un brazo de acceso, común a todas ellas. El brazo puede moverse fácilmente en forma radial a cada una de las posiciones definidas, en cada posición cada cabeza está situada de tal modo que puede leer o escribir en un area en forma de anillo sobre la superficie, llamada pista. El conjunto de todas las pistas en cada posición del brazo se llama cilindro. Cada pista contiene uno o más bloques. Para acceder al bloque, se mueve el brazo hacia el cilindro,se selecciona la cabeza, y se explora la pista hasta que el bloque queda en la posición de la cabeza.

Si el disco contiene registros de longitud variable (Fig.20), se localiza un registro particular explorando primero la pista que contiene el registro hasta que aparece un punto índice, entonces se accede al registro explorando la pista y contando los registros hasta que están separados por espacios, como los bloques en las cintas. La dirección de un registro viene dada por el número de cilíndro, el número de pista y número de registro.

Si el disco contiene registros físicos de longitud fija (Fig.21)-se le llama organizado por sectores- la densidad de grabación varia de las

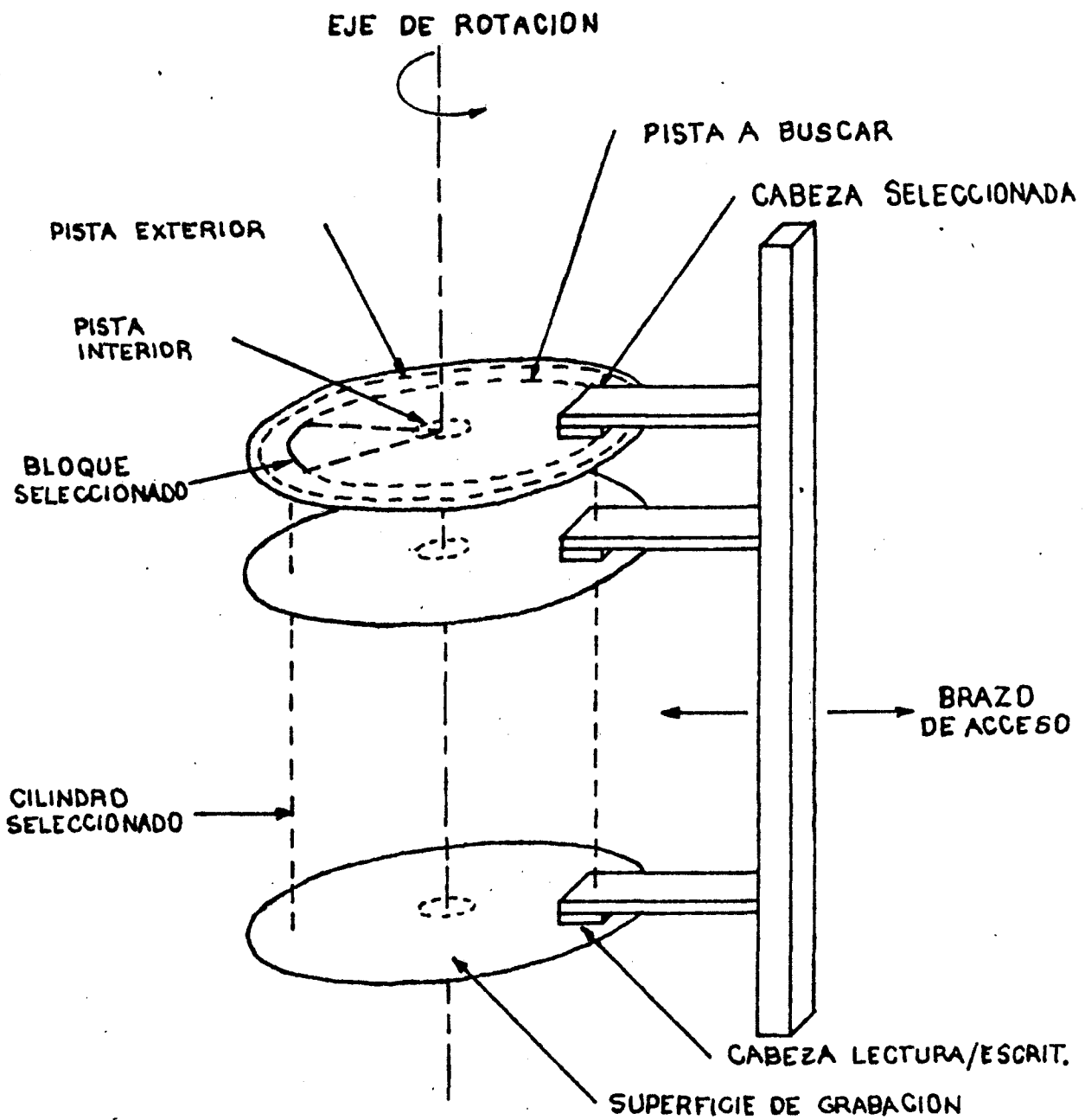
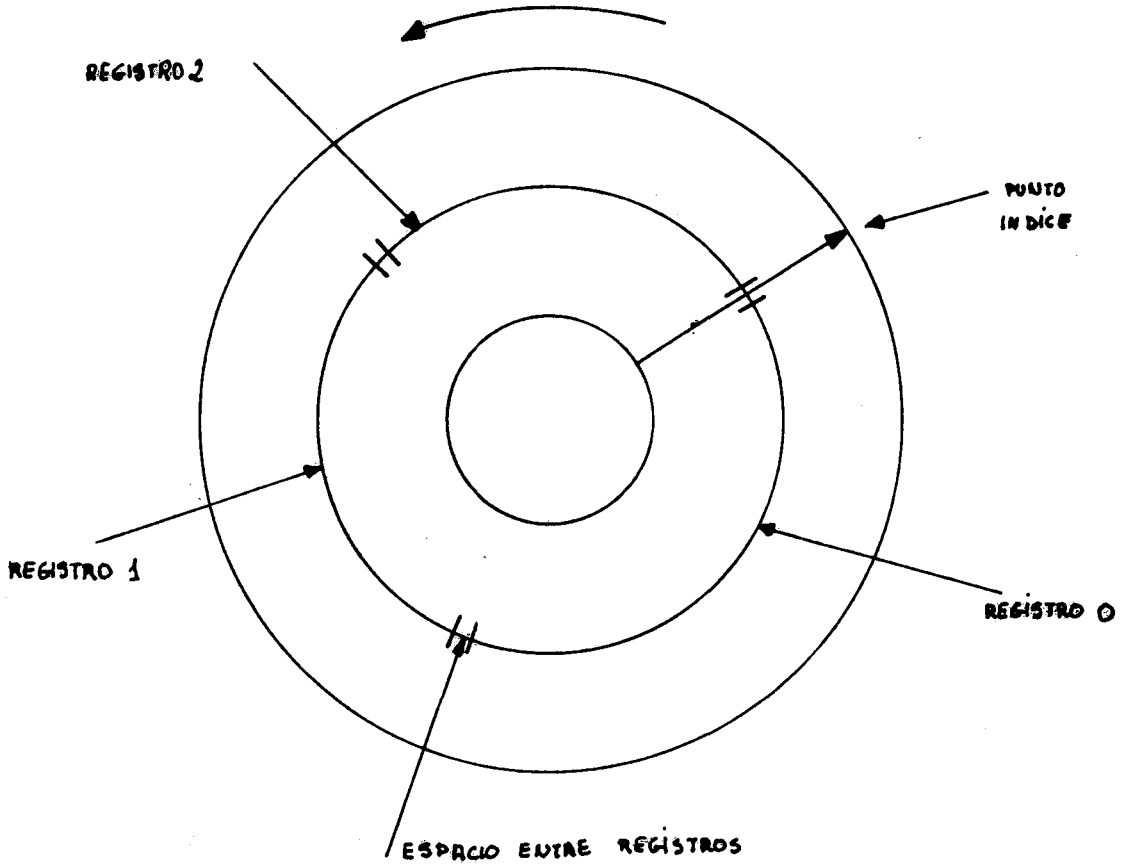


fig.19 unidad de disco de cabeza móvil

fig.20 disco de registros de longitud variable.



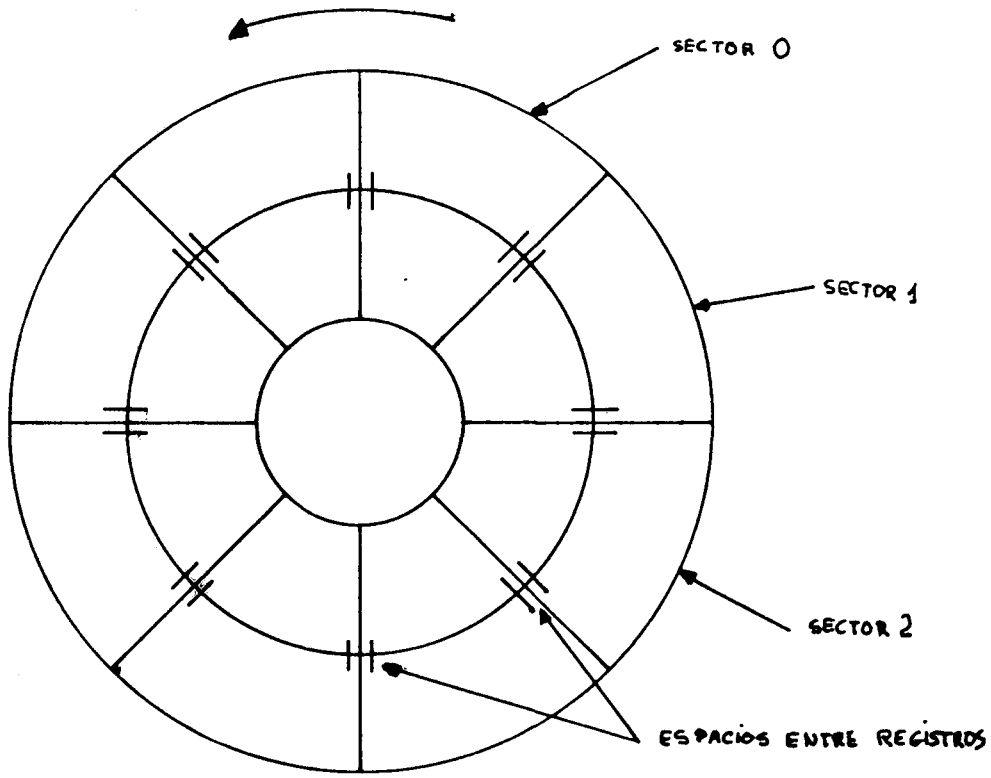


fig.21. disco organizado por sectores

pistas exteriores a las interiores de forma que todas contengan el mismo número de registros físicos. Un registro se direcciona por el número de cilindro, número de pista y número de sector.
Como en las cintas, la unidad de datos que se lee o escribe en el disco se llama bloque. Sin embargo, aquí no coinciden forzosamente con los registros físcos.
Un bloque puede ocupar parte de un registro físico, o varios registros físicos, la mayoría de los discos permiten la lectura y escritura de registros físicos contiguos como una operación continua, de hecho, es posible transferir el contenido de un disco como un bloque. En tal caso, se accede a los registros físicos en secuencia automaticamente. Por ejemplo si un disco tiene "s" sectores, "p" pistas y "c" cilindros, la secuencia será desde el cilindro 0, pista 0, sector 0 hasta el cilindro c-1, pista p-1, sector s-1.
Hay tres operaciones básicas en un disco:
- Seek: (Busqueda) Mover el brazo hasta un cilindro dado y seleccionar la pista especificada.
- Read: Lectura de un bloque empezando en un registro físico dado.
- Write: Escritura de un bloque empezando en un registro físico dado.

Un fichero físico en undisco es una colección dada de registros físicos, no necesairamente contiguos, como era en el caso de las cintas magnéticas. El tiempo total de acceso de lectura/escritura es la suma de:
- Tiempo de posicionamiento de las cabezas.
- Tiempo de localización rotacional dentro de una misma pista.

Existen unidades de discos con un número de cabezas fijas (cabezas por pista) que son mucho más rápidas.
Cada pista del disco tiene una dirección, generalmente el fabricante proporciona un software que nos permite especificar el tamaño de un registro y fichero, luego acceder a un registro específico. Los registros están numerados de 1 a n,donde n es el número de registros del fichero, de este modo podemos tratar un fichero como un número de registros separados numerados sin que impote la dirección física de la pista donde está almacenado el registro.

2.2 FICHEROS FISICOS

En general, la estructura externe de un fichero es la de un conjunto de

registros lógicos. Cada registro lógico contiene datos relacionados en un conjunto de campos. Cada registro lógico contiene el mismo conjuto de campos, o sea, el formato de los campos de cada registro es identico al de los demás. El campo usado para identificar un tegistro se llama el campo de clave.

El nombre fichero lógico se usa para indicar que el programados ve sólo las relaciones lógicas entre los datos del fichero, es decir, ve el fichero como una colección de registros lógicos.

Un fichero físico es el fichero almacenado en la unidad de memoria auxiliar. Los ficheros físicos se dividen en registros físicos, algunas veces, un registro lógico puede formar un registro físico, por ejemplo cada registro lógico de un fichero puede ser del mismo tamaño del de un registro físico. Pero en general no coinciden, así el registro lógico puede ser más pequeño que el registro físico y varios registros lógicos ocuparán un registro físico, a esto se le llama agrupar los registros lógicos. Sin embargo, puede suceder que un registro lógico sea mayor que el registro físico y el resultado se llama "registros seccionados", que son registros lógicos divididos entre uno o más registros físicos.

El tamaño del registro físico (sectores en los discos y bloques en las cintas) viene dado por el fabricante, intervienen cuatro factores en su elección:

- El tiempo de acceso: El tiempo total para acceder a un fichero puede disminuirse si aumentamos el tamaño de los bloques.
- El uso de la memoria auxiliar: En sistemas de registros físicos de longitud variable, el tamaño del bloque determina el tamaño del registro físico. La memoria se puede aprovechar haciendo los bloques mayores.
- El uso de la memoria principal: Bloques grandes precisan de grandes buffers en la memoria principal durante las transferencias, y grandes buffers pueden reducir el uso de la memoria principal.
- El tamaño del registro físico: Un bloque de menor tamaño que un registro físico desaprovecha el restro de la capacidad del registro físico.

CAPITULO III :

TRATAMIENTO DE LA INFORMACION. SISTEMAS DE GESTION DE LA INFORMACION .

3.1 TRATAMIENTO DE LA INFORMACION

Hasta ahora hemos hecho un estudio de los ficheros como sistemas de información según su estructura y la disposición física y lógica de sus elementos.Ahora haremos un estudio desde el punto de vista funcional de los sistemas de gestión de información.
Podemos definir como funciones básicas de un sistema de gestión de información las siguientes:

1. Mantener el control de toda la información del sistema mediante varias tablas - la mayor es el directorio de fichero,también llamado Tabla De Contenidos Del Volumen (U.T.O.C.).

2. Determinar la política para dictaminar dónde y cómo se almacena la información y quién tiene acceso a esa información.

3. La localización de las fuentes de información requeridas para un determinado proceso dentro del sistema.

4. La reincorporación de los recursos cuando ya no son necesarios,cuando la información ya no nos hace falta.

Al conjunto de los módulos de tratamiento de información se le llama Sistema De Fichero. El tratamiento de la información es una de las partes primordiales de un sistema operativo.

Dentro de los sistemas de gestión de información podemos hacer distinción entre:

- El sistema de fichero : Trata sólo de la simple organización lógica de la información,sin estructurar ni interpretar a nivel del sistema operativo.

- El sistema de tratamiento de datos:realiza algunas estructuraciones pero interpretación de la información.Por ejemplo,el sistema puede saber que un fichero se divide en entradas de un cierto tamaño,y aunque permite al programador manipular éstas entradas,no conoce lo que ellas significan (el sistema de tratamiento de datos IMS-II de IBM).

- El sistema de base de datos : Trata tanto con la estructura como con la

1. El directorio básico de ficheros (BFD).
2. La tabla de Ficheros Activos (AFT) .
3. Las comunicaciones con el módulo ACV.

III) VERIFICACION DEL CONTROL DE ACCESO (A.C.V.) : Una llamada típica a éste sistema es :

CALL ACU (READ,2,4,12000)

READ:Función de leer.

2:Entrada de la AFT

4:Numero del registro.

12000:Posición de memoria.

El ACV actúa como punto de comprobación entre el sistema básico de fichero y el sistema lógico de ficheros.
Compara la función deseada (READ, WRITE) con los accesos permitidos que se indican en la entrada de la tabla AFT.
Si no se permite acceso, existe una condición de error y la petición al sistema se anula. Si se permite el acceso,se pasa el control, al sistema lógico de ficheros.

IV) SISTEMA LOGICO DE FICHEROS (LFS) : Es el cuarto paso de procesamiento ,una llamada típica a él sería:

CALL LFS (READ , 2,4,12000)

READ: Función de lectura

2: Entrada a la tabla AFTN

4: Número de registro

12000: Posición de memoria

Que es idéntica en forma a la llamada al ACV (paso III). El LFS convierte la petición para acceder un registró lógico en una petición para una cadena de bytes.O sea, un fichero se trata como una cadena secuencial de bytes (sin formato de registros físicos).
Cuando la longitud del registro es fija (en nuestro caso 500 bytes) la

interpretación de la información.Como ejemplo podemos tomar un sistema de reservade pasajes en unas líneas aéreas,que nos permite preguntar ¿ Cuantas personas vuelan en el vuelo 904 ? Aquí trararemos principalmente los sistemas de ficheros.Los sistemas de ficheros envuelven desde simples mecanismos para mantener control de los programas del sistema a procesos complicados que proporcionan al usuario y al sistema operativo la posibilidad de almacenar,compartir o acceder a la información .

Analizemos primeramente una petición sencilla de información y los pasos necesarios para cumplimentarla.De éste modelo sencillo construiremos un modelo general de un sistema de fichero, y luego veremos como diseñar modelos más flexibles.

3.2 UN SISTEMA DE FICHERO SIMPLE: Consideremos los pasos necesarios para procesar la siguiente sentencia (del tipo de lenguaje PL/I):

READ FILE (ETHEL) RECORD (4) INTO LOCATION (12000)

Es una petición para leer el cuarto registro del fichero llamado ETHEL en la posición de memoria 12000 . Supongamos que el fichero ETHEL está almacenado en disco,como se ve en la Fig. 22 . El fichero ETHEL ocupa la parte sombreada y consta de siete registros lógicos .Cada registro lógico tiene 500 bytes de longitud.El disco consta de 16 bloques físicos de 1000 bytes cada uno. Así podremos almacenar dos registros lógicos del fichero ETHEL en cada bloque físico, en vez de usar la dirección con número de pista y de registro,asignaremos a cada bloque físico un número como en la Fig. 22 Aunque físicamente los bloques que forman el fichero ETHEL están dispuestos en forma contigua,podrían estar esparcidos de forma salteada.

Hay seis entradas en el directorio de la Fig. 23 , tres de ellas (MARILYN:ETHEN Y JOHN) son de ficheros de usuarios.Las otras tres son para tener la cuenta del espacio de almacenamiento no usado.Los bloques 0 y 1 están reservados para propósitos especiales que explicaremos más tarde.

PASOS A REALIZARSE: En la figura 24 vemos los pasos para cumplir la petición de nuestra sentencia:

READ FILE (ETHEL) RECORD (4) INTO LOCATION (12000) '.

Para satisfacerla se han de realizar unos pasos en orden,según un algoritmo y nuestro directorio de fichero elemental.

Podemos sacar un modelo general de sistema de fichero,según los pasos vistos en F.24 y tenemos el organigrama de la figura 25 ,en él cada nivel depende sólo de los que están debajo.Aunque varíen algo,éste es el modelo

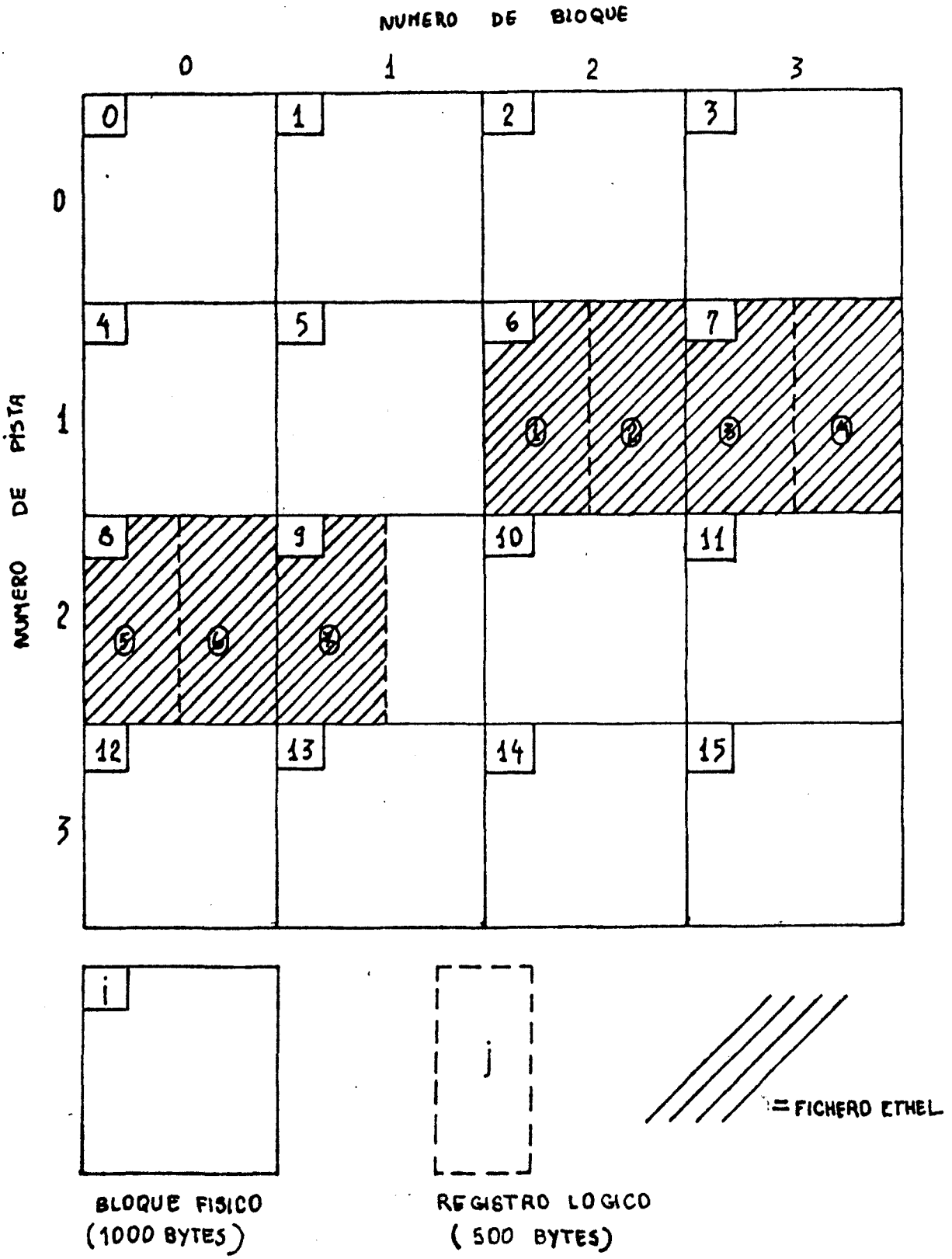


fig.22. almacenamiento físico del fichero ethel

NUMERO DE ENTRADA	NOMBRE	TAMAÑO DEL REG. LOGICO	NUMERO DE REGISTROS LOGICOS	DIRECCION DEL PRIMER BLOQUE FISICO	PROTECCION Y CONTROL DE ACCESO
1	MARILYN	80	10	2	ACCESIBLE POR TODOS
2	LIBRE	1000	3	3	(ZONA LIBRE)
3	ETHEL	500	7	6	SOLO LECTURA
4	JOHN	100	30	12	LECTURA PARA MADNICK LECTURA/ESCRITURA PARA DONOVAN
5	LIBRE	1000	2	10	(ZONA LIBRE)
6	LIBRE	1000	1	15	(ZONA LIBRE)

fig.23 ejemplo de directorio de fichero

PASO NUMERO	ACCION
1	EL DIRECTORIO DEL FICHERO SE EXPLORA PARA ENCONTRAR LA ENTRADA DE ETHEL (NUMERO 3)
2	LA INFORMACION ACERCA DE ETHEL SE EXTRAE, O SEA, TENEMOS: TAMAÑO REGISTRO LOGICO, DIRECCION PRIMER BLOQUE, E INFORMACION DE CENTRAL Y ACCESO.
3	BASADA EN LA INFORMACION DE PROTECCION, SE HACE UNA DESICION SEGUN SEA EL PROCESO PARA ACCEDER A ETHEL.
4	OBTENCION DE LA DIRECCION DEL BYTE LOGICO, MEDIANTE LA FORMULA: DIRECCION BYTE LOGICO = (NUMERO REGISTRO - 1) x x (TAMANO REGISTRO) = (4 - 1) x (500) = 1.500.
5	LA DIRECCION DEL BYTE LOGICO SE TRANSFORMA EN UNA DIRECCION DE BYTE FISICO. LA DIRECCION DEL BYTE LOGICO 1.500 CORRESPONDE AL BYTE 500 DENTRO DEL SEGUNDO BLOQUE FISICO DE ETHEL, QUE CORRESPONDE AL BLOQUE FISICO 7.
6	EL BLOQUE FISICO 7, SE LEE EN UN BUFFER DE MEMORIA PRINCIPAL DE 1.000 BYTES
7	EL REGISTRO 4 SE EXTRAE DEL AREA BUFFER Y PASA AL AREA DEL USUARIO EN LA POSICION 1 2000

fig.24. pasos a desarrollar en una peticion de lectura (READ)

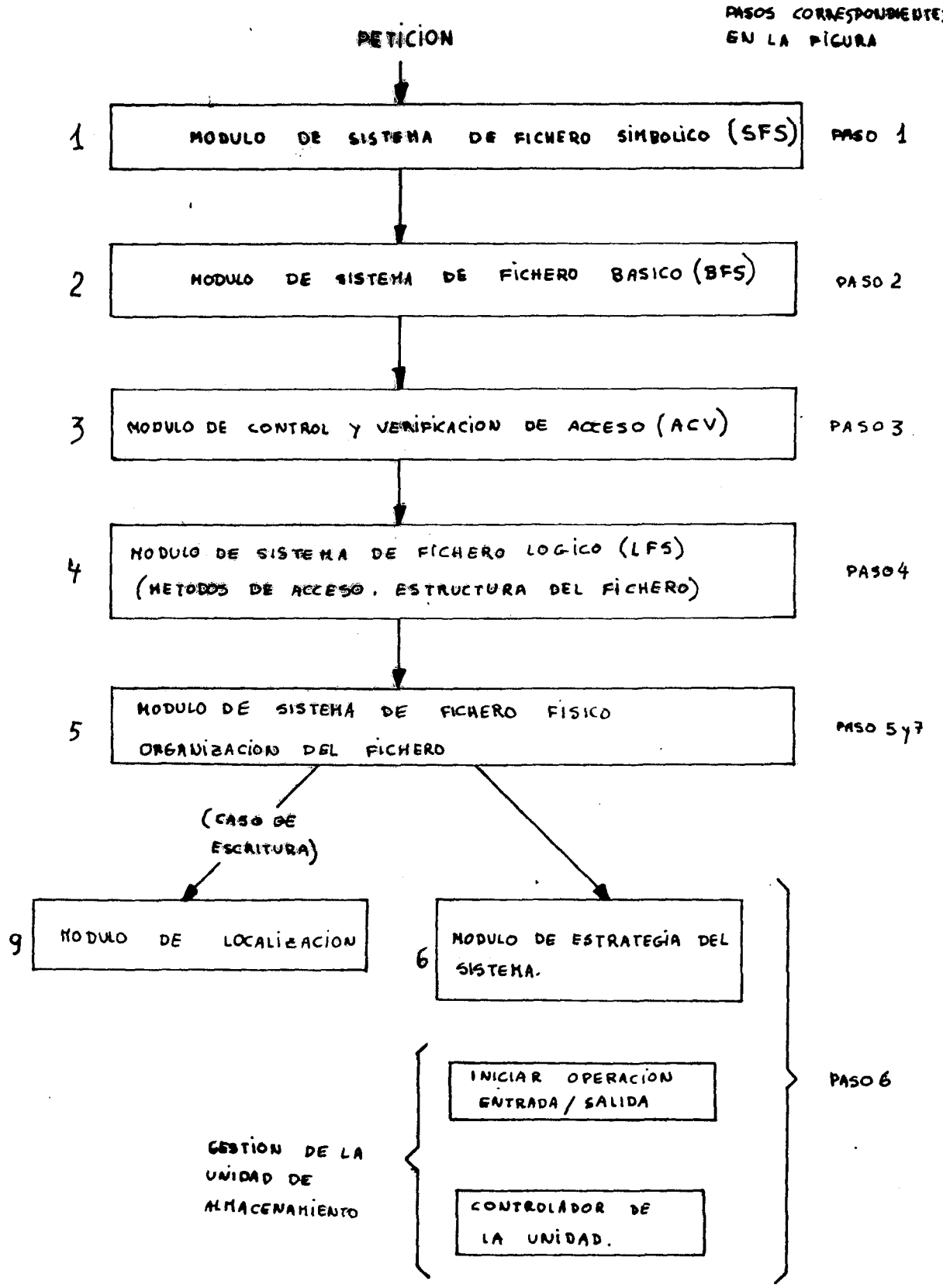


fig.25 modelo jerarquizado de un sistema de fichero.

general en que se basan todos los sistemas de fichero usados hasta ahora.

3.3 MANTENIMIENTO DEL DIRECTORIO DE UN FICHERO: ¿ Cómo se crean y se introducen las entradas en el directorio del fichero? ¿ Dónde se almacena el directorio de fichero?¿ Hay que buscar en todo el directorio cada vez que haya una petición de acceso a fichero? Para responder a éstas preguntas podemos estudiar un ejemplo práctico: el sistema de fichero IBM BOS/360. Dónde se mantien el control del directorio y del espacio de almacenamiento usando sentencias como:

CREATE ETHEL,RECSIZE = 500,NUMRECS = 7 ,LOCATION = 6 DELETE JOHN .

El comando CREATE añade una nueva entrada al directorio,y DELETE nos borra un fichero viejo ya creado.

Para conseguir mayor capacidad en la memoria del sistema,el directorio del fichero se almacena en el soporte de almacenamiento (volumen) además de que así,si vamos a transferir el volumen conteniendo todos los ficheros también transferimos el directorio.Sin embargo,si el directorio es grande, está solución inplica un tiempo largo de búsqueda de la entrada a buscar en cada acceso a fichero,por lo que entonces se suelen copiar en memoria las entradas de los ficheros más corrientemente usados.Hay dos sentencias especiales para ello:

OPEN: Para copiar una entrada específica del directorio de fichero en memoria (abrir el fichero).

CLOSE: La entrada no es ya necesaria en memoria principal (cerrar fichero).

3.4 PARTES DE UN FICHERO : I) SISTEMA SIMBOLICO DEL FICHERO: El primer paso de procesamiento se llama sistema simbolico del fichero (SFS) ,una llamada típica a éste nivel es : CALL SFS (READ, " ETHEL " , 4 , 12000) .

READ : Función de leer en fichero.

ETHEL : Nombre del fichero.

4 : número de registro del fichero a leer.

12000 : posición memoria.

El SFS usa el nombre del fichero para localizar su entrada (exclusiva) en el directorio. En los sistemas simples,a cada fichero le corresponde un nombre exclusivo.Sin embargo existe la posibilidad de tener distintos ficheros con el mismo nombre y viceversa,para ello dividimos el directorio de la

Fig. 23 en dos directorios separados (Fig.26) :
- Un directorio simbólico de ficheros (SFD)
- Un directorio básico de ficheros (BFD)

El sistema simbólico de fichero busca en el SFD para determinar un único identificador para el fichero solicitado, éste identificador se usa para localizar la entrada del fichero en el BFD. Así, la sentencia :

CALL BFS (READ,5,4,12000) : (El número 5 es el identificador de "ETHEL" en el sistema básico de fichero (BFS))

La tabla ANT (tabla de nombres Activos): Se usa para almacenar en ella (en memoria principal) las entradas a ficheros de mayor uso, y así evitar accesos repetidos de entrada/salida al soporte de almacenamiento.
En resúmen, el SFS es el responsable de:

1. El directorio simbólico del fichero (SFD)
2. La tabla de nombres activos (ANT)
3. Las comunicaciones con el sistema básico de fichero (BFS)

II) SISTEMA BASICO DE FICHERO (BFS) : Representa el segundo paso de procesamiento. Una llamada típica a él es:

CALL BFS (READ,5,4,12000)

READ : Función de lectura.

5 : Identificador del fichero.

4 : Número de registro.

12000 : Localización de la memoria.

El sistema básico de fichero usa el identificador del fichero para localizar la entrada en el directorio básico del fichero (BFD) y copia la entrada en memoria principal.
Las entradas para ficheros activos se almacenan en la tabla AFT (Tabla de ficheros activos) . Es la entrada del fichero en la AFT la que se usa en el siguiente paso para la verificación de control de acceso.
En resumen, el sistema básico de fichero es responsable de :

	NOMBRE	IDENTIFICADOR UNICO (ID)
1	MARILYN	3
2	ETHEL	5
3	JOHN	6

16 BYTES — 4 BYTES

ENTRADA DE 20 BYTES

(a) DIRECTORIO SIMBOLICO DE FICHEROS

ID	TAMAÑO DEL REGISTRO LOGICO	NUMERO DE REGISTROS LOGICOS	DIRECCION DEL PRIMER BLOQUE FISICO	PROTECCION Y CONTROL DE ACCESO
1	—	—	0	(DIRECTORIO BASICO DE FICHEROS)
2	20	3	1	(DIRECTORIO SIMBOLICO DE FICHEROS)
3	80	10	2	ACCESIBLE POR TODOS
4	1.000	3	3	(LIBRE)
5	500	7	6	SOLO LECTURA
6	100	30	12	LECTURA POR HADNICK LECTURA/ESCRITURA POR DONOVAN.
7	1000	2	10	(LIBRE)
8	1000	1	15	(LIBRE)

(b) DIRECTORIO BASICO DE FICHEROS

fig.26 ejemplos de directorios basico y simbolico

conversión es :

Dirección del Byte lógico = (Número Registro - 1) Longitud del Registro

Longitud del Byte Lógico = Longitud del Registro.

Estos dos parámetros se usan en vez del número de registro en el siguiente paso, el sistema físico del fichero.

En resúmen el LFS :

- Convierte la petición de registro en perición de una cadena de bytes.

- Hace la comunicación con el sistema físico del fichero.

V) SISTEMA FISICO DE FICHERO (PFS) : Es el quinto paso en el procesamiento de nuestra sentencia, una llamada típica a él es :

CALL PFS (READ , 2,1500,500,12000) .

Read : Función de lectura .

2 : Entrada a AFT.

1500 : Byte de dirección.

500 : Byte de longitud.

El PFS usa el byte de dirección junto con la entrada a AFT para determinar al bloque físico que contiene la cadena de bytes deseada. Este bloque se lee y asigna a un buffer de memoria principal.
Para pasar el byte de Dirección Lógico al número de bloque físico (en nuestra configuración de la Fig. 22) hacemos la operación :

$$\text{N Bloque Físico} = \frac{\text{Byte Dirección Lógica}}{\text{Tamaño Bloque Físico}} + \text{Dirección 1 bloque Físico.}$$

Si la cadena de bytes empieza en el Byte de Dirección Lógica 1500, tenemos.

$$\text{N Bloque físico} = \frac{1500}{1000} + 6 = 1 + 6 = 7$$

Que es el bloque que contiene el registro 4 de ETHEL (que es el que queremos leer).

La localización de la cadena de bytes dentro del bloque físico puede determinarse mediante la igualdad:

$$\text{Comienzo dentro del bloque físico} = \text{RESTO}\left[\frac{\text{Dirección Bloque Lógico}}{\text{Tamaño Bloque Físico}}\right]$$

en nuestro caso:

$$\text{resto}\left[\frac{1500}{1000}\right] = 500$$

O sea, la cadena de bytes empieza en la posición 500 dentro del bloque físico, como veíamos en la figura .

Si todos los soportes de almacenamiento quedan iguales, los tamaños de los bloques y las funciones de direccionamiento podrían ser rutinal fijas dentro del PFS, pero como ésto no ocurre, toda la información necesaria en éste procedimiento se incluye en el propio volumen de almacenamiento y se lee en la llamada Tabla de organización Física (POT) cuando el sistema es inicializado.

En resúmen el PFS realiza lo siguiente :

1. Convierte la petición de cadena lógica de bytes en el número de bloque físico y hace la localización dentro del bloque.

2. Crea la tabla POT . (Tabla de organización física).

3. Hace la comunicación con el modelo de estrategía de localización y el modelo de estrategia del sistema .

VI) MODULO DE ESTRATEGIA DE COLOCACION : (ASM) : Es el encargado de mantener el control de los bloques sin usar de cada unidad de almacenamiento. Una llamada típica es:

CALL ASM (6,4,FIRST BLOCK)

6 : entrada de la tabla POT .

4 : número de bloques.

FIRST BLOCK : primer bloque.

Donde la entrada 6 de la tabla POT corresponde al sistema de almacenamiento en el que se encuentra nuestro fichero ETHEL y la variable FIRST BLOCK representa la dirección del primer de los cuatro bloques.
El ASM puede ser invocado por un procedimiento de organización especial,como el comando CREATE que ya hemos visto.

VII) MODULO DE ESTRATEGIA DEL SISTEMA: (DSM) : Convierte el número de bloque físico en la dirección de formato necesaria del sistema (ej. : BLOQUE FISICO 7 = PISTA 1,REGISTRO 3), realiza los comandos de entrada / salida necesarios para cada tipo de unidad. Todos los sistemas vistos anteriormente eran independientes de la unidad de almacenamiento,son embargo el DSM es dependiente de ella.

VIII) ORGANIZADOR DE ENTRADA / SALIDA Y CONTROLADOR DE LA UNIDAD:Estos módulos corresponden a las rutinas de tratamiento de los sistemas de almacenamiento,realizan la lectura del bloque fisico que contiene la información pedida en nuestra sentencia.

3.5 COMUNICACIONES ENTRE LOS MODULOS DEL SISTEMA DE FICHERO: Una vez hecha la transferencia de entrada / salida física,se devuelve el control al módulo de estrategia del sistema (DSM) .El DSM comprueba que está completa y devuelve el control al PFS,que extrae la informacion deseada del buffer y la coloca en la posición de memoria designada.Un código de " operación cumplida " pasa por todos los otros módulos del sistema de fichero.
Hay 2 razones por las que no pasa el control directamente al nivel más alto (jerarquicamente) :

- Un error puede detectarse en cualquier nivel y ese nivel debe enviar entonces un código de error apropiado.
- Un nivel puede hacer varias llamadas a otros niveles inferiores.

Una función de todos los niveles es la conprobación o verificación.El SFS comprueba para ver si existe o no el fichero a buscar.El ACV comprueba si

el acceso está permitido. El LFS comprueba si la dirección lógica es del fichero pedido.El BSM comprueba si existe o no la unidad.

3.6 DEFECTOS EN EL DISEÑO DE UN SISTEMA DE FICHERO SIMPLE : Resumamos algunos de los inconvenientes que supone nuestro sistema de fichero simple estudiado hasta ahora:

a) En cuanto al sistema físico de fichero (SFS) : Hemos supuesto que acada fichero le corresponde un único nombre ,y viceversa.

b) En cuanto al control de acceso y verificación (ACV): ¿como se comparten los ficheros entre varios usuarios ? ¿ Como puede un mismo fichero tener diferentes atributos de protección para distintos usuarios ?

c) Sistema lógico de fichero (LFS) : Hemos supuesto que cada fichero consiste en registros de longitud fija y secuenciales. ¿ Como se realiza el acceso a registros de longitud variable ?¿ Existen otras estructuras de ficheros?

d) Sistema físico de fichero (PFS) : ¿Existen otras estructuras físicas distintas de la secuencial ? ¿ Seran unas eficientes en cuanto al aprovechamiento del espacio?
Si hemos supuesto que a cada petición este módulo hace un acceso al disco,¿ que sucede si a la petición READ ETHEL RECORD (4) le sigue una petición READ ETHEL RECORD (3) ?, ¿ No estaría ya el registro 3 de ETHEL en el buffer de la memoria ?

e) Estrategia de localización : ¿ Como se almacenan los ficheros en el almacenan los ficheros en el almacenamiento secundario ? ¿ No habrán problemas de fragmentación al borrar un fichero ? ¿ Se usa eficientemente el espacio de memoria de almacenamiento secundario ?

f) Modulo de estrategia del sistema: ¿ Puede éste módulo reestructurar o reordenar las peticiones para usar con más eficiencia las catacterísticas de la unidad de almacenamiento ?

En las secciones siguientes veremos téctincas más generales para cada uno de

éstos apartados,que darán al usuario más flexibilidad y aumentarán la eficiencia del sistema.

3.7 SISTEMA SIMBOLICO DE FICHERO : Como hemos visto, su función es pasar la regerencia simbólica del usuario (nombre del fichero) a un identificador.

Veamos las formas de ampliación de nuestro sistema simbólico de fichero:

a) Varios nombres asignados al mismo fichero : (propiedad llamada " alias "), es muy útil cuando hay varios programas que dan al mismo fichero distintos nombres, o cuando el usuario, según las circunstancias usan uno u otro nombre según su significado.

b) Ficheros compartidos: (enlazados) : El acceso a un mismo fichero por varios usuarios da mayor eficacia y flexibilidad al sistema.Los buffers de la memoria principal,así como el espacio de almacenamiento secundario pueden compartirse.Sólo hace falta una copia del fichero,aunque hayan varios accesos a él.

c) Mismo nombre a diferentes ficheros : Puede pasar que varios usuarios den el mismo nombre a distintos ficheros (por ejemplo TEST) si tenemos los comandos :

DELETE TEST o CREATE TEST

Podemos estar borrando el fichero de otro usuario.Mantener un control manual para evitar duplicar nombres no es la solución más eficaz.

d) Subdivisión de ficheros en grupos : Ficheros referentes a distintas tareas pueden agruparse en grupos para mantener mejor control sobre ellos , a éstos grupos se les llama librerías.

Un sistema de nombramiento de ficheros más flexible que el que hemos estudiado presentaría la posibilidad de incluir éstas posibilidades.

- Los Ficheros Directorio: En nuestro fichero elemental hemos separado el directorio de fichero en dos al hacer nuestro estudio: - Directorio simbólico de Ficheros (nombre del fichero)
  - Directorio básico de Ficheros (fichero físico)

En general podemos hacer una generalización y tenemos :

- Varios directorios simbólicos (varios usuarios)
- Tratar a su vez éstos directorios simbólicos como ficheros.

O sea,que cada programador puede tener asignado un directorio simbólico diferente, y luego hay un directorio maestro que especifica la entrada al directorio básico de ficheros para cada directorio simbólico particular.

3.8 EL SISTEMA BASICO DE FICHERO (BFS) : Con el identificador propio del fichero a buscar el BFS busca en la tabla AFT (tabla de ficheros activos) pará ver si el fichero está abierto,si no,va al directorio básico de ficheros para encontrar la entrada del fichero y la pasa a la tabla AFT.
Cuando se comparte un fichero entre varios usuarios (está en más de un directorio simbólico) para que el programador sepa que éste fichero aparece más de una vez,hay un contador en la entrada al directorio Básico de fichero que indica al número de entradas de directorios simbólicos de ficheros (para cada usuario) que se refieren a él,cuando se da una sentencia por parte del programador de borrar ese fichero de su directorio,el contador se decrementará en uno.

3.9 VERIFICACION Y CONTROL DE ACCESO : La facilidad de compartir los ficheros,aporta una serie de ventajas a nuestro sistema:

1. Ahorra la duplicación de ficheros.Por ejemplo,si tenemos un fichero compartido en un disco,que ocupa 128 K ,cuando quitemos el fichero a ser compartido,alrededor de 50 copias (privadas) para cada usuario aparecerán ocupando 6.4 bytes.

2. Permite la sincronización en las operaciones con datos.Si dos programas están actualizando blances de un banco,es conveniente que se refieran al mismo fichero exactamente.

3. Mejora el rendimiento del sistema por ejemplo,si la información se comparte en la memoria,hay una reducción en las operaciones de entrada/salida y accesos a ficheros.

- Existen tres maneras de controlar los derechos de acceso a ficheros:

a) Matriz de control de Acceso : Partimos del hecho de que el ordenador es capaz de identificar a cada uno de los usuarios que están realizando una tarea.
La matriz de control de Acceso es una matrizde 2 dimensiones en la que se almacena:

- En las filas : la lista de todos los ficheros del sistema.
- En las columnas : todos los usuarios del ordenador.

Así,cada elemento de la matriz,según usuario y fichero,permite o no el acceso según el módulo de Verificación y control de Acceso conprueba que se cumple el permiso de acceso del usuario al fichero (ver Fig. 27)
Cuando hay muchos ficheros y ususarios se adoptan soluciones que sean más sencillas, como la Lista de control de Acceso que acompaña a cada fichero y que indica que usuarios tienen acceso a él,y con el grupo ALL OTHERS generalizamos el resto de los usuarios,por ejemplo:

FICHERO ACCESS CONTROL LIST, (ACL)

DATA DONOVAN (ALL ACCESS),MADNICK (READ)
CARPENTER (READ),ALL OTHERS (NO ACCESS)

b) Claves o Contraseñas : La lista y la matriz de control de acceso tienen el problema de ocupar gran cantidad de espacio,y su modificación puede ser complicada por su gran número de entradas.
Otro método es usar las claves,que el usuario debe dar al hacer la petición a un fichero dado para tener acceso a él.Este método tiene la ventaja de que se necesita poco espacio para la información de protección del fichero,pero tiene la desventaja de que el programador puede tener acceso a la clave,pues ésta se encuentra almacenada en el sistema.

c) Criptografía : Se trata de codificar los ficheros criptograficamente, todos los usuarios pueden acceder a los ficheros codificados,pero sólo el usuario autorizado sabe la manera de decodificar el contenido.
La decodificación (lectura) y codificación (escritura) se hace por medio del módulo de Verificación y Control.El usuario da la clave del código (argumento),que puede cambiar cuando lo desee.
La ventaja de éste sistema frente a los dos anteriores es que la clave

Usuarios

Ficheros

	donovan	madnick	pedro	john
sqrt	lectura	lectura/ escritura	lectura	no tiene acceso
pl/i	no tiene acceso	lectura	escritura	no tiene acceso
stock price	lectura	lectura	no tiene acceso	lectura/ escritura
pay roll	escritura	lectura/ escritura	lectura	no tiene acceso

fig.27. matriz de control de acceso

del código no se almacena en el sistema, sólo hay que introducirla cuando codificamos ó decodificamos un fichero.

3.10 SISTEMA LOGICO DE FICHEROS : Es el que concierne con el trazado de la estructura de los registros lógicos como cadenas de bytes dentro del sistema físico.
En los sistemas de ficheros, se pueden dar varias estructuras de registros, llamadas métodos de acceso :

I) Registros de longitud fija y estructurados secuencialmente :

Pueden tener acceso secuencial o directo.

II) Registros de longitud variable y estructurados secuencialmente:

También pueden tener acceso secuencial o directo, pero aquí hemos de jugar con una nueva variable, que es la longitud del registro.

III) Registros con clave estructurados secuencialmente:

Se basan en el uso de una insignia o clave, y se ordenan secuencialmente según el orden establecido de éstas claves.

IV) Registros con varios indicativos :

Cuando se quiere que los datos sean localizados por más de una referencia (ejemplo, clasificar los libros por autor, materia, título).

V) Registros de estructura eneadenada:

Proporcionan mayor flexibilidad en la estructura de datos complejos.

VI) Registros de estructura relacionada :

Los registros están relacionados unos con otros según su información.

3.11 SISTEMA FISICO DE FICHEROS : Como hemos visto, la función de éste módulo es pasar la dirección del byte lógico en dirección de bloque físico (del soporte). Aparte de lo ya visto en el fichero elemental, vamos a ver tres características:

1. Minimización de las operaciones de entrada / salida :

Recordemos nuestro fichero ETHEL (7 registros lógicos) si lo quisieramos leer secuencialmente ,esto requeriría 7 operaciones de entrada / salida,pero podemos reducir éste número sabiendo que el fichero entero ocupa sólo 4 bloques físicos y como tenemos un control sobre los bloques físicos,podemos hacer que en cada operación de lectura un bloque físico se copie en la memoria,así cuando queramos leer el primer registro de ETHEL llevamos el bloque físico 6 al buffer de la memoria,luego,la petición de lectura del registro segundo no precisa de una nueva operación de EIS sino que se operará sobre el buffer de la memoria,(recordar que en nuestro sistema,cada bloque físico contiene dos registros lógicos).

2. Posibilidad de hacer el tamaño del registro lógico independiente del tamaño del bloque físico :

En el sistema que hemos estudiado un registro físico debe mantener un número fijo de registros lógicos,sin embargo puede ocurrir que la longitud del registro lógico no sea una fracción entera de la longitud del registro físico,existen varias maneras de solucionar éste problema,como dejar espacios en blanco que sean el margen de diferencia entre las longitudes de los registros y de los bloques,pero ésto no es un sistema muy efectivo.
Otra forma es variar el formato de las pistas de almacenamiento,posibilidad que permite el sistema IBM 2314 y 3330,pero en general,no es muy conveniente ésta solución.
Una solución más general es modificar los pasos 5 y 7 de nuestro algoritmo (Figura 24- 18) y permitir así a los registros lógicos distribuirse mejor en los bloques físicos,también nos permtie almacenar registros que sean de mayor longitud que los bloques.
Figura

3. Posibilidad de disposición del Fichero de Forma no contígua:

No siempre es posible distribuir el fichero en bloques físicamente contíguos,entonces se usan generalmente dos tecnicas:

a. Bloques Encadenados : En éste modo,cada bloque físico contiene la dirección del próximo bloque contiguo en la estructura lógica.La entrada del sistema básico de fichero contiene la dirección del primer bloque físico ,pero luego las siguientes vienen dadas por los"punteros" .

Estos punteros se almacenan dentro del bloque físico,o bien como parte del campo de encabezamiento del bloque.Los bloques encadenados son eficientes cuando el acceso es secuencial pero no cuando el acceso es directo.

b. Mapa del Fichero : Otra manera es usar una tabla de Trazado del fichero que asigna a cada dirección de bloque lógico su dirección de bloque físico.Este " mapa del fichero " se almacena como parte de la entrada en el BFD en un bloque aparte.Cuando se abre el fichero,se copia el mapa en la tabla AFT.

3.12 MODULO DE ESTRATEGIA DE LOCALIZACION: En algunos sistemas,el programador ha de controlar el espacio y la disposición de almacenamiento del fichero y pueden haber errores de solapamiento de ficheros,por lo que es mejor que el programador unicamente cree o borre ficheros y dejar la disposición de éstos al sistema.Así,tenemos varias técnicas :

- Localización automática : En el directorio se especifican las áreas libres especificandolas con entradas " free " ,así cuando hay una solicitud de 4 bloques para ETHEL el sistema busca una entrada en el directorio que sea libre y de 4 o más bloques de longitud.

- Localización dinámica : El modo de localización automática visto tiene dos problemas:

a. Induce a la fragmentación en el almacenamiento secundario.
b. El usuario no suele conocer la longitud del fichero que va a almacenar.

Para usar el espacio de almacenamiento secundario mejor,se puede hacer de otra manera:

- Relocalización (hacer compactos los ficheros de un disco periódicamente)
- Localización del espacio del fichero no contigua (permite la fragmentación y facilmente la ampliación del fichero).

En ésto consiste la localización dinámica.

- Mapas de fichero libre : La localización automática presenta el problema de que si en vez de tener pocas áreas libres de gran tamaño,tenemos muchas áreas pequeñas,el directorio de ficheros se hace muy grande.Entonces,una solución es usar un mapa de fichero que con-

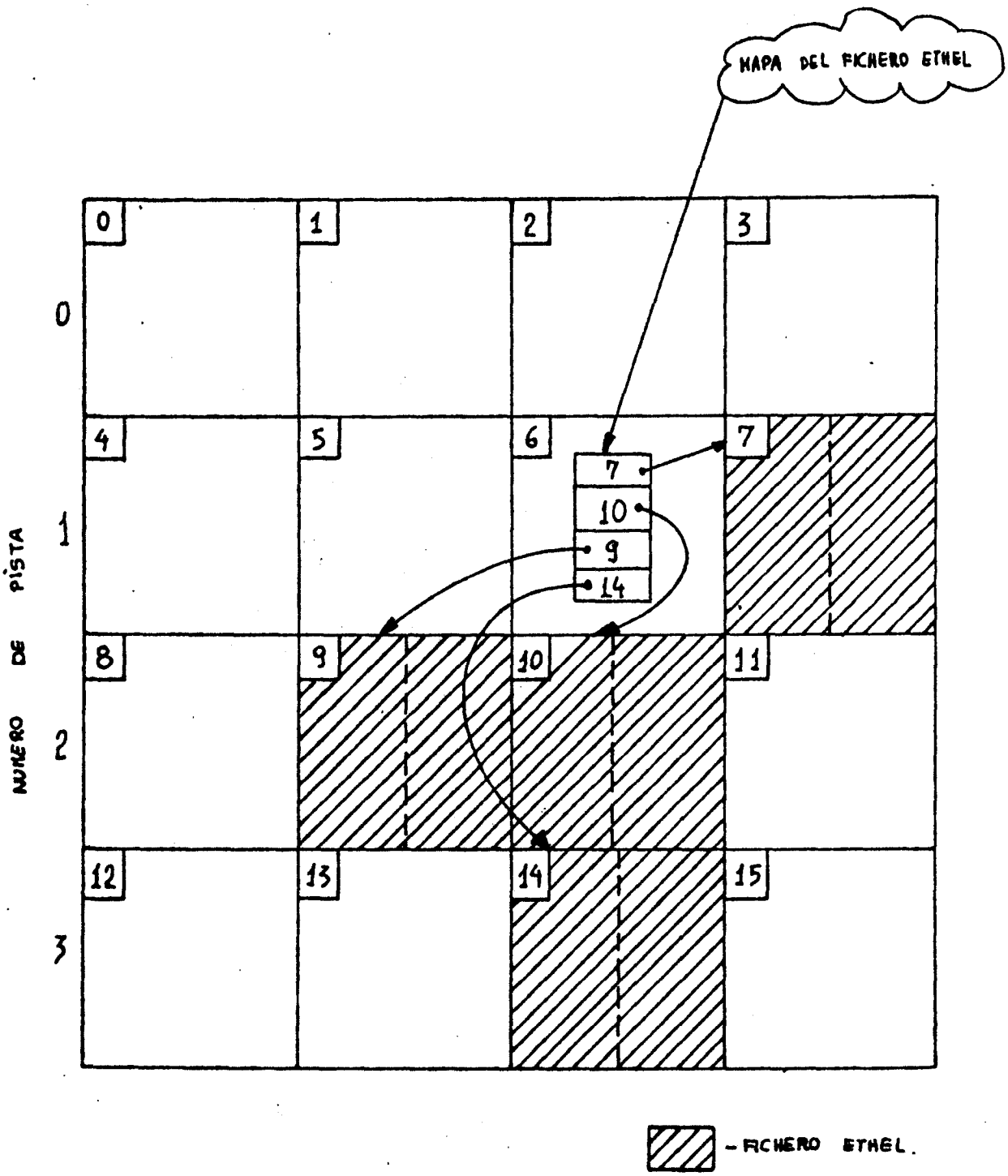


fig.28. mapa de fichero

tenga la lista de todos los bloques disponibles.Cuando se utiliza un bloque se elimina del mapa y cuando queda libre se inserta en la lista.

- Encadenamiento libre : Otra forma es encadenar todos los bloques libres, así cuando se necesita un bloque se toma del principio de la cadena,y cuando no se usa se coloca de nuevo en ella.

- Mapa de Bits : Otro sistema es usar una secuencia de bits que indique el estado de los bloques,así ,para 16 bloques tenemos la palabra de 16 bits :

1111110000100000

1 : bloque correspondiente en uso

0 : bloque libre

En ésta palabra tenemos reflejado el estado de los 16 bloques.

3.13 MODULO DE ESTRATEGIA DEL SISTEMA : Sus funciones son,como sabemos :

1. Pasar la dirección física a dirección de la unidad de almacenamiento (el registro 4 de ETHEL es el bloque 7,pista 1,grabación 3).

2. Crear el programa canal,que haga lo siguiente:

   - buscar pista 1
   - explorar grabación 3 en la pista 1
   - esperar a la localización
   - leer

3. Petición de entrada/salida,o sea,realizar la transferencia de información

4. Manejar las interrupciones de entrada / salida y las condiciones de error que puedan ocurrir.

CAPITULO IV :

ESTUDIO DE LOS SISTEMAS DE GESTION DE DATOS DEL ORDENADOR VAX DE DIGITAL.

4.1 SISTEMAS DE GESTION DE FICHEROS DEL VAX - 11

Como aplicación práctica del estudio de las estructuras y la gestión de ficheros veremos como funciona el servicio de gestión de datos del sistema operativo del ordenador VAX de DIGITAL.

Entre los servicios disponibles del sistema de tratamiento tenemos:

- Tratamiento de datos y ficheros
- Sistema de ficheros
- Servicio de mantenimiento de registros
- Controladores de las unidados de almacenamiento
- Interprete de los comandos

Entre las utilidades que el VAX ofrece tenemos el VAX - 11 SORT/ MERGE para reordenamiento de datos, el DATATRIEVE para reordenamiento de datos y escritura de informes, y el FMS para hacer formatos con pantalla.

El sistema de ficheros provee la estructuración de los volúmenes (contenedores de la información) y el acceso del directorio a los ficheros de disco y de cinta.

El programador puede usar el sistema de ficheros como base para construirse su propio sistema de procesamiento de registros, o puede usar los servicios del sistema de gestión de registros VAX/VMS.

Los servicios de gestión de tegistros (Record Management Services o RMS) nos proveen un acceso a todos los periféricos de entrada/salida (de todos los tipos) independiente del tipo de unidad de almacenamiento. Los procesos del RMS permiten a nuestros programas acceder a los registros de los ficheros y sirven de interface independientemente de las características de la unidad.

Los controladores de las unidades nos proporcionan el manejo básico de las unidades de entrada/salida para todo el resto de los sevicios de gestión de datos.

El interpréte de los comandos permite al usuario reservar unidades para su uso exclusivo y asignar nombres lógicos a las especificaciones de los ficheros.

El intérprete de los comandos también permite al usuario efectuar operaciones de convertir, borrar y copiar ficheros.

4.2 TRATAMIENTO DE FICHEROS:

El sistema VAX/VMS proporciona dos estructuras de ficheros: una para almacenamiento en disco y otra para almacenamiento en cimta. Desde el punto de vista del usuario, las únicas diferencias entre las dos estructuras de ficheros son las impuestas por las caracteristicas del medio de almacnamiento.

La protección de ficheros se basa en los códigos de Identificación del usuario (User Identification Codes o UICs).

Los UICs establecen la relación del que accede a la información según sea como propietario, grupo de propietarios, el sistema, o el mundo (todo el resto de usuarios). Según sea esta relación, el que accede podrá o no podrá leer, escribir, ejecutar o borrar un determinado fichero.

Los volúmenes de almacenamiento en disco son para en uso de narios usuarios. Contienen una jerarquía de directorios a varios niceles que se define por los usuarios del volúmen.

Los volúmenes de almacenamiento en cinta son de uso para un solo usuario.

Los bloques están contiguos físicos, el agrupamiento de los bloques está hecho bajo control de programa.

DIRECTORIOS DE FICHEROS Y SUS ESTRUCTURAS:

Como hemos visto, un fichero directorio contiene en cada entrada a un fichero dado, su nombre, tipo, versión e identificador único del mismo.

Un volúmen de almacenamiento de disco contiene al menos un directorio, llamado el directorio maestro de ficheros. El manager del sistema es el responsable de crear el directorio maestro del volúmen, este directorio maestro contendrá una lista de los directorios de fichero que forman un segundo nivel de directorios. Este segundo nivel de directorio es el fomado por los que nos listan los fichero de datos y también por otros fichero directorio.

A estos directorios los llamamos subdirectorios, el usuario puede crear subdirectorios dentro de los directorios que posee.

ESPECIFICACIONES DE FICHEROS:

Una especificación de fichero identifica que fichero vamos a usar en una operación de procesamiento de ficheros. Una especificación de fichero completa es una cadena de caracteres que se compone de los siguientes campos:

- Nombre del Nudo (Node Name): se refiere al nombre del nudo de la red de almacenamiento al que está almacenado el volúmen que contiene el fichero. Va seguido de dos dobles puntos (::) para diferenciarlo de la especificación.
- Nombre de la unidad: Es el nombre de la unidad en la que se encuentra el vonúmen que contiene al fichero en cuestión. El nombre de la unidad va seguido de dos puntos (:) para difereciarlo del resto.
- Nombre del directorio: Es el nombre del directorio en el que está listado el fichero, comienza con un corchete ([ ) y acaba con otro corchete ( ]).
- Nombre del fichero: Es el nombre del fichero asignado al usuario.
- Tipo de fichero : Es la identificación del tipo del fichero, va precedido por un punto (.).
- Versión del fichero : Es el número de la versión del fichero con que trabajemos, va precedido de un punto (.) o punto y coma (;).

Por ejemplo, una especificación completa de un fichero es:

NODE 47 :: DBA1 :[JONES] HANO I. FOR; 2

En este caso el NODE 47 (nudo 47) es el nombre del nudo de la red, DBA 1 es el nombre de la unidad (DB por el disco, A por controlador de disco, 1 por el número de la undad), JONES es el nombre del directorio, HANOI es el nombre del fichero, FOR es el tipo de fichero (en este caso es un fichero FORTRAN) y el 2 es el número de la versión.

No es necesario dar una especificación completa para identificar un fichero. Por ejemplo, si el nombre del nudo no figura, se supone que es el nudo en el que estamos ejecutando nuestro programa. Si el número de versión no figura, se supone siempre que es la última versión. El nombre de la unedad y del directorio los define el manager del sistema dentro de el fichero de autorización de usuarios.

NOMBRES LOGICOS DE FICHEROS:

El usuario y sus programas no están limitados a identificar los fiche-

ros con las especificaciones de ficheros vistas. Pueden usar los nombres lógicos en lugar de una especificación completa del fichero, o bien en lugar de una parte de la especificación. Por ejemplo, el usuario puede asignar un nombre lógico a los tres campos de la izquierda de la especificación mediante la sentencia:

$ ASSIGN NODE 47 :: DBA 4 : [JONES] to VOL

Entonces, podemos usar el nombre lógico VOL en una sentencia que precisa de la especificación del fichero, como el comando TYPE:

TYPE VOL: HANOI. FOR

TRATAMIENTO DE FICHERO:

El sistema VAX/VMS inxluye varios servicios que sirven de ayuda en el tratamiento y el mantenimiento de datos. Veamos algunos de ellos:

- Ordenamiento de ficheros: El programa SORT/MERGE (Ordenamiento/ Unión) permite al usuario arreglar, modificar o borrar los registros de un fichero.
- Comparación de ficheros: Mediante un comando de comparación podemos contrastar dos ficheros alineando automáticamente los textos a igualar, y el resultado es una lista de las diferencias entre los ficheros.
- Almacenamiento preventivo de ficheros: Gracias a el procedimiento de guardar y comprimir en disco (Disk Save and Compres o DSC) el usuario puede guardar el comtenido total del disco en cinta magnética o en otros discos, para una recuperación de la información caso pérdida.
- Verificación de estructuras de ficheros: La verificación de ficheros nos sirve para comprobar la consistencia y exactirud de la estructura de un fichero. Nos puede mostrar también el número de bloques disponible en un volúmen, localiza ficheros a los que no se pondría acceder de otro modo, y nos da una lista de los nombres de los ficheros del volúmen.
- Utilidades del RMS: Los servicios de tratamiento de registros (Record Management Services o RMS) se complementan con un número de servicios diseñados especialmente para la creación y mantenimiento de ficheros RMS. Dan la posibilidad al usuario de, entre otras cosas:

a) Crear un fichero RMS y definir sus atributos.
b) Hacer una lista de los atributos de un fichero o grupo de ficheros.
c) Convertir un fichero de un determinado formato de registros en otro

fichero con otro formato distinto.

4.3 SERVICIOS DE MANTENIMIENTO DE REGISTROS:

Son una serie de procesos del sistema que nos proporcionan las facilidades para el almacenamiento, recuperación y modificación de datos.
La forma en que los servicios de mantenimiento de registro (RMS) crean un fichero nos da lo que llamamos la organización de este fichero. Los servicios RMS crean tres tipos de organización diferentes:

- Secuencial (los registros están en orden consecutivo).
- Relativa (células de registros de logitud fija).
- Indexada (los registros poseen una clave).

La organización de un fichero establece la técnica a usar en los procesos de leer y escribir datos en un fichero, a estas tecnicas se les llama modos de acceso a los regis tros, y los modos de acceso que proporcionan los servicios RMS son:

- Secuencial (los registros se leen o escriven en la secuencia que establece la organización del fichero).
- Aleatorio (el programa establece el orden en que se procesan los registros).
- Por direccionamiento del registro de un fichero (sirve para recuperar los registros de cualquier tipo de organización de fichero).

4.4 ATRIBUTOS DE REGISTROS Y FICHEROS:

Cuando se crea un fichero RMS, o bien el programa o bien el usuario define las características lógicas y físicas del fichero, llamadas atributos.
El usuario define el nombre del fichero, el código de protección y de identificación, y elige el tipo de organización del fichero.
Además de esto, el usuario define otros atributos, como son:

- Unidad de almacenamiento
- Tamaño del fichero
- Localización del fichero
- Tamaño y formato del registro
- Claves (en el caso de fichero indexado)

De este modo, el usuario puede crear un fichero según sus requerimientos y darle las características que precise al fichero.

- Encontrar un registro. Se localiza un registro dado dentro de un fichero y obtenemos la información de donde se encuentra (su posición dentro del fichero).
- Borrar un registro. Se extrae un registro dado del fichero, esto no es posible en la organización de ficheros del tipo secuencial.
- Acrualizar un registro. El programa modifica el contrnido de un registro del fichero, esto no es posible en ficheros de organización secuencial (excepto para los de disco).

4.5 LENGUAJES DE UTILIDAD EN EL TRATAMIENTO DE FICHEROS:

El sistema operativo del VAX/VMS proporciona una serie de facilidades para el tratamiento de datos, como son: DATATRIEVE, VAX - 11 SORT y el sistema de tratamiento de formas (Forms Management System o FMS).

a) DATATRIEVE: Es un conjunto de elementos de software que proporciona acceso fácil, rápido y directo a los datos contenidos en los ficheros. Está diseñado para que el usuario no tenga que conocer a fondo todo el sistema de gestión, y su uso permite al usuario no experto en programación tener acceso a todo el proceso de tratamiento de datos.

Veamos algunos de los comandos básicos usados para acceder, modificar y mostrar datos:

- HELP: Proporciona un resumen de cada comando perteneciente al sistema DATATRIEVE.
- SORT: Reordena una serie de registros en una secuencia ascendente o bien descendente según los contenidos de uno o más campos de los registros.
- PRINT: Muestra uno o más campos de uno o más registros, bien listándolos en impresora o bien pasandolos a un fichero de disco.
- MODIFY: Modifica los valores de uno o más campos de un registro o grupo de registros.
- STORE: Crea un nuevo registro, el contenido de cada campo del registro es dado por el usuario.
- ERASE: Borra uno o más registros de un fichero RMS.
- FOR: Ejecuta un comando una vez para cada registro en un grupo de registros, como si fuera una estructura en bucle.

El sistema DATATRIEVE se provee de una serie de operadores aritméticos (suma, resta, multiplicación, división y negación), operadores estadís-

ticos (total, media, máximo, mínimo y cuenta) y realiza la conversión automática entre los tipos de datos usados en los lenguajes FORTRAN, COBOL, DIBOL y BASIC.

b) VAX - 11 SORT/MERGE:

Mediante los procedimientos de SORT el usuario puede reordenar los datos de un fichero - considerado como un fichero de entrada - y crear un nuevo fichero ordenado en un secuencia, basada en una clave, de los registros del fichero de entrada. El usuario puede especificar hasta 10 ficheros de entrada y el proceso SORT producirá un fichero (ordenado) como resultado.

La secuencia de ordenamiento la determinan los campos de control especificados por el usuario (llamados campos de clave); y el fichero ordenado servirá para tener los datos ordenados según la clave elegida y acceder a ellos en ses orden.

El programa MERGE permite al usuario unir desde 2 hasta 10 ficheros que estén ordenados de forma similar y dar como el resultado un fichero, el MERGE une los datos de acuerdo con los campos de clave definidos por el usuario.

Los ficheros de entrada deben estar ordenados, o sea que los campos de clave de los procesos SORT y MERGE deben ser iguales cuando vayamos a ordenar y unir varios ficheros en uno solo.

Los procesos SORT y MERGE pueden usarse como una seire de subrutinas llamadas desde un programa, por ejemplo, se pueden llamar desde programas en COBOL (VAX - 11 COBOL) usando las sentencias COBOL, SORT y MERGE.

c) SISTEMA DE TRATAMIENTO DE FORMATOS (FMS):

Los procesos del sistema FMS del VAX - 11 sirven de soporte para la creación de formatos en pantalla de video para los terminales VT100, y proporcionan un interface flexible y entre el usuario del terminal y el programa de aplicación de formatos.

CAPITULO V :

USO DEL SISTEMA OPERATIVO ISIS-II PARA EL CONTROL DE LA UNIDAD DE DISCOS FLOPPY.

5.1 CONTROL DE LA UNIDAD DE DISCOS DEL SISTEMA DE DESARROLLO

Como parte práctica en el estudio de ficheros, haremos el estudio de como manejar la unidad de discos y sus ficheros a través de un programa en lenguaje ensamblador. Para ello se hace uso del sistema operativo ISIS-II, el monitor, y de su acceso atravéś de otros programas.
Las rutinas de ISIS-II y del monitor que pueden utilizarse por llamada en un programa de usuario incluyen las siguientes:

- Operaciones de entrada/salida para el disco y los periféricos standard del Intellec, excepto el programador Universal de PROM. Como son: OPEN (abrir el fichero), CLOSE (cerrar un fichero), READ (lectura del contenido de un fichero), WRITE (escritura en un fichero), SEEK (colocación del marcados de fichero), RESCAN (colocar el marcador al principio de la linea), SPATH (obtención de la información relativa a un fichero).
- Mantenimiento del directorio del disco, ATTRIB, DELETE, RENAME.
- Asignamiento de la unidad de consola y salida de los mensajes de error, CONSOL, WHOCON, ERROR.
- Carga y ejecución de un programa y retorno al supervisor, LOAD, EXIT.
- Rutinas del monitor para controlar los sistemas periféricos, CI, CO, RI, PO, LO, UI, UO.
- Rutinas del monitor para extender el sistema de entrada/salida a controladores definidos por el usuario, IODEF.

El acceso a estas rutinas en una llamada al ISIS-II que especifica la rutinas deseadas y la dirección de la lista de parámetros para las rutinas.
Para clasificar el efecto de ciertas llamadas de sistema en los ficheros, dos cantidades, LENGTH (longitud) y MARKER (marcador), se asocian a cada fichero en su descripción. LENGTH es el número de bytes del fichero y MARKER es el número de bytes acabados de leer o escritos en el fichero (esto es, actúa como un punteo de fichero).

5.2SINTAXIS DE LLAMADAS DE SISTEMA

Muchas de las llamadas de sistemas de ISIS-II tienen nombres y funciones similares a los de el ISIS-II (refieranse al manual de usuario del ISIS-II). Esto es así porque el uso de ISIS-II por otro programa es esencialmente lo mismo que el uso del ISIS-II cuando estamos manejando la consola. Las llamadas de sistema al ISIS-II pueden ser usadas por el lenguaje ensamblador o el PL/M. Cuando tengamos un programa con llamadas de sistemas al ISIS -II, deberemos linkar el programa con la libreria

de sistema SISTEM. LIB usando el programa LINK.

El SYSTEM. LIB es un fichero libreria suministrado en el disco de sistema. Contiene todo el procedimiento necesario para servir de interface entre el programa del usuario - con las llamadas de sistema - con el sistema operativo ISIS-II.

5.3 LLAMADAS EN LENGUAJE ENSAMBLADOR

El interface entre el programa en ensamblador y el ISIS-II se realiza llamando a un único punto de entrada etiquetado ISIS-II, y pasando dos parámetros. El primer parámetro es un número que identifica la llamada de sistema; el segundo es la dirección de un bloque de control que contiene los parámetros adicionales requeridos por la llamada de sistema. El primer parámetro se pasa en el registro C y la dirección del bloque de control se pasa en el registro DE. El punto de entrada debe definirse en el programa del usuario con la sentencia:

```
EXTRN ISIS
```

El punto de entrada ISIS está definido en una rutina del SYSTEM. LIB que debe incluirse en el programa del usuario. Los números identificadores de llamadas deben definirse con sentencias EQU antes de que el programa usuario haga referencia a ellos. Solo las llamadas específicas a las rutinas necesarias de nuestro programa necesitan ser definidas. La siguiente tabla lista los números identificadores para las llmadas de sistema:

SYSTEM CALL	IDENTIFICADOR
OPEN	0
CLOSE	1
DELETE	2
READ	3
WRITE	4
SEEK	5
LOAD	6
RENAME	7
CONSOL	8
EXIT	9
ATTRIB	10
RESCAN	11
ERROR	12
WHOCON	13
SPATH	14

5.4 LLAMADAS A ENTRADA/SALIDA DE FICHERO

Seis llamadas de sistema están disponibles a nuestro programa para controlar la entrada/salida a fichero.

Estas subrutinas nos permiten abrir un fichero para operaciones de lectura o escritura, mover el puntero de un fichero abierto, y cerrar los ficheros cuando hemos acabado. Estas rutinas del supervisor transfieren bloques de longitud variable de datos entre los periféricos standard y un area de memoria buffer de nuestro programa. Además del buffer de transferencia de datos de nuestro programa, el supervisor de disco necesita dos buffers de 128 bytes para cada fichero de disco abierto.

5.5 PRECAUCIONES DE LAS LLAMADAS DE SISTEMA

Como algunas llamadas de sistema del ISIS-II a su vez realizan llamadas a rutinas del monitor, no se deben mezclar llamadas a rutinas del sistema, ya que pueden ocurrir resultados no predecibles.

Cuando hay una interrupción, la posición de ejecución del programa se almacena. Si enviamos una llamada de sistema como parte de una rutina de servicio de interrupción, la información de la posición se perderá, con resultados impredicibles.

Un buen programa debe tener frecuentes llamadas de comprobación del byte de estado (STATUS) cuando hagamos frecuentes llamadas de sistema.

Pasemos a ver las rutinas de gestión de ficheros del sistema operativo ISIS-II:

- OPEN: Inicializa el fichero para operaciones de entrada/salida.

La llamada OPEN inicializa las tablas del ISIS y localiza los buffers que se necesitan para el procesamiento de entrad/salida del fichero especificado.

Deberemos dar una lista de cinco variables con la llamada OOEN, que son:

a) La dirección del campo de 2 bytes en el que ISIS almacenará el número de tabla del fichero activado (AFTN), esto es el número identificador que usará el ISIS en sus talas del fichero que acabamos de abrir. Nuestro programa hará uso también de este valor para otras llamadas relativas a este fichero.

b) La direción de la cadena de caracteres ASCII que contiene el nombre del fichero a abrir. Esta cadena debe cumplir con los requisitos de nombre de fichero impuestos por el sistema operativo ISIS.

c) Un valor indicando el modo de acceso para el que el fichero ha sido abierto. Si este valor es un 1 quiere decir que el fivhero ha sido abierto solo para entrada al sistema, o sea, para lectura (READ). Si este valor es 2, quiere decir que el fichero abierto es sólo para salida des sistema

(WRITE). Se es 3 es que el fichero ha sido abierto para modificaciones (lectura y escritura). Cuando un fichero se abre para lectura, el marcador se pone a cero. Se el fichero especificado para lectura no existe, el sistema da un error. Cuando se abre el fichero para escritura, el marcador y la longitud se ponen a cero. Si el fichero especificado para salida no existe, se crea un nuevo fichero de disco con el nombre especificado. Cuando abrimos un fichero para modificar, el marcador se pone a cero. Si el fichero ya existe, su logitud no variará, si no existe, se crea un nuevo fichero de disco como en el caso de escritura. Abrir un fichero en un modo de acceso que es físicamente imposible, como por ejemplo abrir el fichero impresora : LP : para lectura, da como resultado un error.

d) El AFTN del fichero eco en el caso de que el fichero se abre para edición en linea. Este fichero eco ha sido abierto previamento para escritura (ACCESS = 2). Un fichero es editado en linea cuando el usuario quiere tener la posibilidad de corregir sus errores al escribir una linea en teclado, así, la tecla rubout y los controles le permiten hacer las correcciones y entonces transmitir una línea perfecta apretando la tecla de retorno del carro.

El AFTN del fichero eco (fichero que nos muestra lo que escribimos) se pasa en el byte menos significativo del campo. Si este campo contiene un cero, no se realiza la edición de línea.

e) La dirección de una posición de memoria para el retorno de números de error, caso producirse errores en la llamada o los parámetros.

Veamos un ejemplo de programa en lenguaje ensamblador con una llamada de sistema OPEN:

```
        EXTRN   ISIS    ;LINKADO CON ISIS
OPEN    EQU     Ø       ;IDENTIFICADOR DE LLAMADA
        MVI     C,OPEN  ;CARGA DE IDENTIFICADOR
        LXI     D,OBLK  ;DIRECCION BLOQUE PARAMETROS
        CALL    ISIS    ;LLAMADA A SISTEMA  OPERATIVO
        LDA     STAT    ;CARGA BYTE STATUS
        ORA     A
        JNZ     ERROR   ;SALTO A RUTINA DE ERROR
OBLK;                   ;BLOQUE PARAMETROS OPEN
        DW      OAFT    ;DIRECCION AFTN FICHERO A
                        ;ABRIR
        DW      OFILE   ;DIRECCION NOMBRE FICHERO
```

```
ACCES:    DW    1      ;EL ACCESO A FICHERO ES  PARA
                       ;LECTURA
ECO:      DW    Ø      ;NO HAY FICHERO ECO
          DW    STAT   ;DIRECCION BYTE STATUS
OAFT;     DS    2      ;AFTN DEL FICHERO (RETORNADO)
STAT;     DS    2      ;BYTE STATUS (RETORNADO)
OFILE:    DB    ":FØ:JOSE.SRC" ;NOMBRE FICHERO A ABRIR
                       ;EN EL NOMBRE DEL FICHERO
                       ;PODEMOS ESPECIFICAR EL DRIVER
                       ;EN QUE LO QUEREMOS ABRIR .
                       ;ESTOS VALORES DE LOS PARAMETROS
                       ;SON VALORES PARA UN EJEMPLO DE
                       ;UN FICHERO A ABRIR PARA HACER UNA
                       ;LECTURA EN EL.
                       ;
```

- READ: Transferencia de datos del fichero a la memoria.

La llamada READ transfiere los datos de un fichero abierto a una posición de memoria especificada por nuestro programa. Con esta llamada debemos incluir una lista de cinco variables:

a) El número AFTN del fichero precisamente abierto para lectura o modificación, este valor ha sido devuelto en una operación de llamada a OPEN precedente en nestro programa, y es 1 para la entrada por consola (CI).

b) La dirección de un buffer que contiene los datos leídos del fichero abierto, este buffer debe ser al menos tan largo como el valor de la cuenta del apartado"c". Si el buffer es muy chico, las posiciones de memoria que siguen al buffer se llenarán con el contenido del fichero.

c) El número de bytes - o cuenta - que se van a transferir del fichero al buffer (COUNT).

d) La dirección de una posición de memoria en la que ISIS almacenará el número real de bytes transferidos del fichero a la memoria (ACTUAL). Este número se le añade al marcador del fichero, y nunca será mayor que el número especificado en cuenta - variable del apartado "c" -. Si el fichero no es editado en línea, el nímero de bytes es igual o bien a la cuenta o a la longitud menos el marcador, que siempre será menor. Si la cuenta en cero, entonces se ACTUAL el cero podemos o no estar en el fin de fichero.

El fin fichero queda mejor indicado en el caso de ficheros editados en linea y la cuenta mayor que cero por el valor ACTUAL igual a cero; y en el caso de ficheros no editados en linea y la cuenta mayor que cero por ACTUAL menor que la cuenta, es decir se transmitió el último bloque.
e) La dirección de una posición de memoria en la que el sistema operativo ISIS-II almacenará, caso de producirse, el número de error.
Veamos un ejemplo de programa con la llamada READ:

```
            EXTRN     ISIS
READ        EQU       3          ;IDENTIFICADOR LLAMADA READ

            MVI       C,READ     ;CARGAR IDENTIFICADOR
            LXI       D,RBLK     ;DIRECCION BLOQUE DE
                                 ;PARAMETROS.
            CALL      ISIS
            LDA       RSTAT      ;CARGA BYTE STATUS
            ORA       A
            JNZ       ERROR      ;SALTO SI STATUS≠ Ø.

RBLK;                            ;BLOQUE PARAMETROS LECTURA
RAFT:       DS        2          ; AFTN DEL FICHERO
            DW        BUFFER     ;DIRECCION BUFFER LECTURA
CONLE:      DW        128        ;Nº BYTES A LEER DEL
                                 ;FICHERO
            DW        ACTUAL     ;DIRECCION DE ACTUAL
            DW        RSTAT      ;DIRECCION BYTE STATUS

ACTUAL:     DS        2          ;Nº BYTES LEIDOS DEL
                                 ;FICHERO.(RETORNADO)
                                 ;BYTE STATUS (RETORNADO)

BUFFER:      DS       128        ;BUFFER DE 128 BYTES
                                 ;
```

- Write: Transferencia de datos de la memoria al fichero.
La llamada WRITE transfiere los datos de una posición de memoria especifica llamada buffer a un fichero abierto. Es necesaria una lista de cuatro variables con cada llamada a la rutina WRITE:

a) El número AFTN del fichero abierto para escritura o modificación, este número ha sido devuelto en una llamada OPEN anterior en nuestro programa.
b) La dirección de la posición de memoria del buffer del que los datos se van a transferir al fichero, o una cadena literal de caracteres con el formato .STRING LITERAL, donde el periodo (.) especifica el contenido del buffer etiquetado con .STRING LITERAL.
c) El número de bytes - o cuenta - a transferir del buffer al fichero de salida. Este valor se le suma al marcador del fichero, y si sucede que como resultado el valor del marcador es mayor que la longitud del fichero, entonces se hace la longitud igual al marcador. El número de bytes que se transmiten en un proceso de escritura - WRITE - es exactamente igual a la cuenta. O sea, que si la longitud del buffer es menor que la cuenta, las posiciones de memoria contiguas al buffer serán escritas al fichero.
d) La dirección de la posición de memoria en la que se retornará el número de error, caso de producirse. Veamos el ejemplo de un programa en ensamblador que usa la llamada WRITE:

```
        EXTRN   ISIS    ;LINKADO A ISIS
WRITE   EQU     4       ;IDENTIFICADOR
                        ;LLAMADA WRITE

        MVI     C,WRITE ;CARGA IDENTIFICADOR
        LXI     D,WBLK  ;DIRECCIO BLOQUE
                        ;PARAMETROS
        CALL    ISIS
        LDA     STAT    ;CARGA BYTE STATUS
        ORA     A
        JNZ     ERROR   ;SALTO SI NO ES Ø
                        ;A RUTINA ERROR

WBLK:                   ;BLOQUE PARAMETROS
WAFT:   DS      2       ;AFTN DEL FICHERO
                        ;A ESCRIBIR
```

```
            DW      BUFFER  ;DIRECCION BUFFER
CONESC:     DW      128     ;CONTADOR BYTES A
                            ;ESCRIBIR EN EL
                            ;FICHERO
            DW      STAT    ;DIRECCION STATUS
STAT:       DS      2       ;BYTE DE STATUS

BUFFER:     DS      128     ;BUFFER DE MEMORIA
                            ;DE 128 BYTES
                            ;DE LONGITUD
```

- SEEK: Posicionamiento del marcador de fichero de disco.

La llamada SEEK permite a nuestro programa encontrar la posición o cambiar el valor del marcador del fichero abierto para lectura o modificación. La posición del marcador puede variarse en cuatro modos distintos: hacia adelante, hacia atrás, hacia una posición especifica y al final del fichero. Se produce un error cuando usamos la rutina SEEK con un fichero abierto para escritura.

Necesitamos cinco variables con cada llamada SEEK:

a) El número AFTN del fichero de disco abierto para lectura o modificaciones, este valor ha sido devuelto por una llamada OPEN precedente en nuestro programa.

b) Un valor que va del cero al cuatro y que indica el tipo de acción a ejecutarse sobre el marker.

Los bloques de bloque y byte (descritos más adelante) se usan para repretar la posición actual del marcador, o bien para calcular el desplazamiento deseado.

Si el valor del modo de operación es cero, el sistema devuelve los valores del bloque y byte que nos dan la posición del marcador. Por ejemplo, si el marcador está justo después del primer bloque del fichero, el sistema dará los valores 1 y 0 en las direcciones asignadas a bloque y byte respectivamente. 0 también esta posición la podemos dar con los valores 0 y 128 respectivamente, que apuntan al mismo byte del fichero. El valor

del marcador viene dado según la siguiente ecuación:

MARKER = 128 + (NUMERO DE BLOQUES) + Nº DE BYTE

Si el valor del modo es 1, el marcador se mueve hacia atrás, hacia el principio del fichero. Los valores de bloque y byte definen el desplazamiento, por ejemplo, si el valor de bloque es 0 y el de bytes es igual a 328, el marcador se mueve hacia atrás 328 bytes. Para hallar un desplazamiento de N en general, deben usarse valores de bloque y byte como los sistemas siguientes:

N = 128 + (NUMERO DE BLOQUE) + NUMERO DE BYTE

Si N es mayor que el marcador, o sea, que la ejecución de la rutina SEEK hace que el marcador vaya más atras que el principio del fichero, el marcador se pone a cero (principio de fichero) y se produce un error.
Si el valor del modo es Z, el marcador se mueve a una posición especifica del fichero, entonces los parámetros de bloque y byte definen la posición, por ejemplo, si el bloque es igual a 27 y el byte es igual a 63, el marcador se moverá al bloque 27, byte 63. Si tanto el bloque cono el byte son iguales a cero, el marcador se mueve al principio de fichero. Si el fichero lo hemos abierto para modificar y hacemos que el marcador vaya más alla del fin de fichero, se añaden caracteres nulos en ASCII (OOHH) para extender la logitud del fichero hasta el marcador, y entonces la longitud del fichero se hace igual a la posición del marcador.
Si el valor del modo es 3, el marcador se mueve hacia adelante, hacia el de fichero. Los parámetros de bloque y byte definen el desplazamiento de un valor N, que podemos hallar mediante la ecuación:

N + 128 + (NUMERO DE BLOQUE) + NUMERO DE BYTE

Si el fichero lo hemos abierto para modificarlo y el desplazamiento coloca el marcador más allá del fin de fichero, se añaden caracteres nulos en ASCII (OOHH) para extender la longitud del fichero hasta el marcador, haciendose la longitud del fichero igual a la posición del marcador.
(Si en la ejecución de un SEEK se produce un overglow en la capacidad del disco, se produce un error que nos detiene el proceso).
Por último si el valor del modo es 4, el marcador se mueve hasta el fin de fichero (los valores de bloque y byte se ignoran).
Además de esta variable, nos hacen falta otras tres con la llamada SEEK, que son:
c) La dirección de una posición de memoria que contenga el valor (2 bytes)

del número de bloque. (Un bloque equivale a 128 bytes).
d) La dirección de una posición de memoria que contenga un valor (2 bytes) usado para el número de byte (este número puede ser mayor que 128).
e) La dirección de la posición de memoria en que el ISIS mandará el número de error, caso de producirse.
Veamos un ejemplo de programa con una llamada a rutina SEEK:

```
            EXTRN     ISIS
SEEK        EQU       5       ;IDENTIFICADOR DE
                              ;LLAMADA

            MVI       C,SEEK;CARGA IDENTIFICADOR
            LXI       D,SBLK;DIRECCION DEL BLOQUE
                              ;DE PARAMETROS

            CALL      ISIS    ;
             LDA      STAT    ;CARGA BYTE STATUS
            ORA       A
            JNZ       ERROR   ;SALTO A RUTINA DE
                              ;ERROR
SBLK:                         ;BLOQUE PARAMETROS
                              ;PARA RUTINA SEEK
SAFT:        DS       2       ;AFTN DEL FICHERO
MODO:        DS       2       ;MODO DE FUNCIONA-
                              ;MIENTO DE SEEK
             DW       BLKS    ;DIRECCION DE BLKS
             DW       NBYTE   ;DIRECCION DE NBYTE

BLKS:        DS       2       ;NUMERO DE BLOQUES DE
                              ;DESPLAZAMIENTO
NBYTE:       DS       2       ;NUMERO DE BYTES DE
                              ;DESPLAZAMIENTO
STAT:        DS       2       ;BYTE DE STATUS
```

- CLOSE: Fin de las operaciones de entrada/salida de un fichero.

La llamada de sistema CLOSE nos quita el fichero de las tablas de entrada/salida y borra los buffers localizados por la sentencia OPEN. Debemos cerrar todos los ficheros una vez finalizadas todas las operaciones de entrada/salida.

Deberemos incluir dos variables con cada llamada CLOSE:

a) El número AFTN del fichero a cerrar, este número ha sido devuelto por una llamada OPEN precedente en nuestro programa.

b) La dirección de una posición de memoria para el envío de el número de error, caso de producirse.

Veamos un ejemplo de programa que usa una llamada CLOSE:

```
        EXTRN           ISIS
CLOSE   EQU             1       ;IDENTIFICADOR

        MVI             C,CLOSE;CARGA IDENTIFICADOR
        LXI             D,CBLK  ;DIRECCION DEL BLOQUE
                                ;DE PARAMETROS

        CALL            ISIS
        LDA             STAT    ;CARGA BYTE STATUS
        ORA             A
        JNZ             ERROR   ;SALTO SI NO ES CERO
                                ;A ERROR

CBLK:                           ;BLOQUE DE PARAMETROS
                                ;PARA RUTINA CLOSE
CAFT:   DS              2       ;AFTN DEL FICHERO
        DW              STAT    ;DIRECCION BYTE
                                ;STATUS

STAT:   DS              2       ;BYTE DE STATUS
```

- SPATH: Obtención de la información relativa a un fichero.

La llamada SPATH permite a nuestro programa obtener información relativa

a un fichero especificado. La información obtenida incluye el número de unidad, nombre del fichero y extensión (si la hubiere), tipo de unidad, y si es un fichero de disco el tipo del driver (controlador).
Debemos incluir tres variables con cada llamada SPATH:
a) La dirección de una cadena de caracteres ASCII que contenga el nombre de fichero del que solicitamos la información.
b) La dirección de una posición de memoria (un buffer de 12 bytes) en la que el sistema devolviera nos pondrá la información. Una vez que la llamada ha sido completada, el buffer contendrá la siguiente información:

BYTE 0: Número de la unidad en la que está el fichero
BYTES 1 AL 6: Nombre del fichero
BYTES 7 AL 9: Extensión del nombre de fichero
BYTE 10: Tipo del periférico al que está asociado el fichero
BYTE 11: Tipo del driver (si es una unidad de disco)

Los posibles valores para el número de unidad son:

0 - Disk drive 0
1 - Disk drive 1
2 - Disk drive 2
3 - Disk drive 3
4 - Disk drive 4
5 - Disk drive 5
6 - Entrada por teletipo
7 - Salida por teletipo
8 - Entrada por CRT
9 - Salida por CRT
10 - Entrada por consola de usuario
11 - Salida por consola de usuario
12 - Lector de cinta de papel (por teletipo)
13 - Lector de cinta de papel de alta velocidad
14 - Lector de usuario (1)
15 - Lector de usuario (2)
16 - Perforada de cinta de papel (por teletipo)
17 - Perforadora de cinta de papel de alta velocidad.
18 - Perforadora de usuario (1)
19 - Perforadora de usuario (2)

20 - Impresora de linea
21 - Lista de usuario
22 - Byte bucket
23 - Entrda por consola
24 - Salida por consola
25 - Disk drive 6
26 - Disk drive 7
27 - Disk drive 8
28 - Disk drive 9

El nombre del fichero y la extensión son el nombre del fichero nuestro. El tipo de unidad especifica el tipo de periférico al que pertenece el fichero en cuestión. Los posibles valores son:

0 - Sistema de entrada secuencial
1 - Sistema de salida secuencial
2 - Sistema de entrada/salida secuencial
3 - Sistema de entrada/salida de acceso directo

El tipo de driver nos dice el tipo de controlador, caso de que el tipo de unidad sea el 3 (si en cualquier otro tipo, elte campo queda indefinido), los posibles valores son:

0 - Controlador no presente
1 - Doble cara doble densidad
2 - Doble cara simple densidad
3 - Simple densidad, intregrada (al que corresponde el driver: FØ: del sistema de desarrollo)
4 - Disco duro

c) La dirección de una posición de memoria para que el sistema devuelva el número de error, caso de producirse.

Veamos un ejemplo de programa con llamada SPATH:

```
          EXTRN     ISIS
SPATH     EQU       14          ;IDENTIFICADOR LLAMADA

          MVI       C,SPATH     ;CARGA IDENTIFICADOR
          LXI       D,SBLK      ;DIRECCION BLOQUE PA-
                                ;RAMETROS DE SPATH
          CALL      ISIS
```

```
                LDA     STAT    ;CARGA BYTE STATUS
                ORA      A
                JNZ     ERROR   ;SALTO A ERROR SI NO
                                ;ES CERO

SBLK:                           ;BLOQUE DE PARAMETROS
                                ;PARA RUTINA SPATH
                DW      FILEN   ;DIRECCION NOMBRE FICHERO

                DW      BUFFER  ;DIRECCION DEL BUFFER

                DW      STAT    ;DIRECCION BYTE STAUS

FILEN:          DS      15      ;CAMPO DEL NOMBRE DEL
                                ;FICHERO
BUFFER          DS      12      ;BUFFER DE DATOS
STAT            DS      2       ;BYTE DE STAUS
```

Ahora veamos tres llamadas de sistema que nos permiten cambiar la información del directorio del disco por programa. Podemos borrar, cambiar de nombre, y cambiar los atributos de un fichero de disco.

- DELETE: Borrar un fichero del directorio del disco.

La llamada DELETE nos borra un fichero especificado del disco. El espacio ocupado por el fichero queda libre y puede usarse por otro fichero.

Debemos incluir dos variables en la llamada DELETE:

a) La dirección de una cadena de caracteres ASCII que nos da el nombre del fichero a borrar, este fichero no podrá estar abierto.

b) La dirección de una posición de memoria para el retorno del número de error, caso de producirse.

Veamos un programa de ejemplo con una llamada a DELETE:

```
                EXTRN   ISIS
DELETE          EQU             ;IDENTIFICADOR DE
                                ;LLAMADA
                MVI     C,DELETE
                LXI     D,DBLK
                CALL    ISIS
```

```
                LDA     STATUS      ;CARGA BYTE DE STATUS
                ORA     A
                JNZ     ERROR       ;SI EL BYTE DE STAUS NO ES
                                    ;CERO SALTAR A ERROR

DBLK:                               ;BLOQUE DE PARAMETROS PARA DELETE
                DW      DFILE       ;DIRECCION DEL NOMBRE
                                    ;DEL FICHERO A BORRAR
                DW      STATUS      ;DIRECCION DEL BYTE STATUS
STATUS:         DS      2           ;BYTE STATUS
DFILE:          DB      "CREDIT"    ;NOMBRE DEL FICHERO
                                    ;QUE SE VA A BORRAR
```

El espacio ocupado por el fichero queda libre al colcarsele a los bloques del fichero borrado un indicativo de borrado. De este modo cuando haga falta escribir un nuevo fichero, todos los sectores ocupados por el fichero que hemos borrado estan disponibles para ser ocupados por otros ficheros que escribamos en el disco.

- RENAME: Cambiar el nombre de un fichero de disco.

La llamada RENAME nos permite cambiar el nombre de un fichero por otro. Debemos incluir tres variables junto con la llamada RENAME:

a) La dirección de la cadena de caracters ASCII que contiene el antiguo nombre del fichero.

b) La dirección de la cadena de caracters ASCII que contiene el nuevo nombre del fichero.

c) La dirección de la posición de memoria para el retorno del número de error, caso de producirse.

Veamos un programa de ejemplo con la llamada a la rutina RENAME:

```
            EXTRN   .ISIS
RENAME      EQU     7       ;IDENTIFICADOR

            MVI     C,RENAME ;CARGA IDENTI-
                             ;FICADOR.
            LXI     D,NBLK  ;DIRECCION BLOQUE
                             ;PARAMETROS
            CALL    ISIS
            LDA     STAT    ;CARGA BYTE STATUS
            ORA     A       ;
            JNZ     ERROR   ;SALTO A ERROR SI NO
                             ;ES CERO

NBLK:                        ;BLOQUE DE PARAMETROS
            DW      FICH2   ;DIRECCION ANTIGUO NOMBRE
                             ;DEL FICHERO
            DW      FICH1.  ;DIRECCION NUEVO NOMBRE
                             ;DEL FICHERO
            DW      STAT    ;DIRECCION BYTE STATUS
STAT:       DS      2       ;BYTE STAUS
FICH1:      DB      "NUEVO";NUEVO NOMBRE
FICH2:      DB      "VIEJO";ANTIGUO NOMBRE
```

- ATTRIB: Cambiar los atributos de un fichero de disco.

Nos permite variar por ptrograma los atributos de un fichero determinado.

Con la llamada a ATTRIB debemos incluir cuatro variables:

a) La dirección de una cadena de caracteres ASCII con el nombre del fichero cuyos atirbutos vamos a variar.

b) Un identificador indicando que atributo vamos a cambiar. Puede valer:

0 - Atributo de Invisibilidad

1 - Atributo de sistema

2 - Atributo de protección contra escritura

3 - Atributo de formato

c) Un valor que nos indica si el atributo elejido lo vamos a activar o a desactivar, este valor se almacena en el bit menos significativo del byte menos significativo. Si este valor es 1 quiere decir que el atributo es activado y si es 0 es que el atributo es desactivado.

d) La dirección de una posición de memoria para el retorno del número de error, caso de producirse.

Veamos un ejemplo de programa con la llamada ATTRIB:

```
          EXTRN   ISIS
ATTRIB    EQU     1Ø        ;IDENTIFICADOR

          MVI     A,ATTRIB;CARGA IDENTIFICADOR
          LXI     D,ABLK    ;DIRECCION BLOQUE
                            ;PARAMETROS
          CALL    ISIS
          LDA     STAT      ;CARGA BYTE STATUS
          ORA     A
          JNZ     ERROR     ;SALTO A ERROR SI NO
                            ;ES CERO
ABLK:                       ;BLOQUE PARAMETROS
          DW      NFICH     ;DIRECCION NONBRE FIC.
          DW      2         ;IDENTIFICADOR DE ATRI-
                            ;BUTO ( WRITE PROTECT )
          DW      1         ;ACTIVACION PROTECCION
          DW      STAT      ;DIRECCION BYTE STATUS
NFICH:    DB      "CREDIT";NOMBRE DEL FICHERO
STAT:     DS      2         ;BYTE DE STATUS
```

- ERROR: Mandar un mensaje de error a la consola del sistema.

La llamada error nos permite enviar un mensaje de error, en caso de que se produzca, a la consola.

Mediante esta rutina, y en combinación con cualquiera de las otras, podemos ver que errores hemos cometido al hacer una llamada a las rutinas de sistema que hemos visto. Según sea el número de este error, y mirando la lista de mensajes de error en el apéndice C de la guía de usuario del ISIS-II, podremos saber que error hemos cometido en el programa.

Denemos incluir dos variables con la llamada a rutina de error:

a) El número de error a ser mandado a la consola, este número debe estar en los 8 bits menos significativos del parámetro. Solamente los números del 101 al 199 inclusive deben usarse para los programas del usuario, el resto - del 0 al 100 y del 200 al 255 - está reservado a los programas de sistema. El sistema muestra el error en el siguiente formato:

ERROR nnn, USER PC mmmm

Donde nnn es el número de error especificado en la llamada y mmmm es la dirección de retorno a nuestro programa.

b) La dirección de una posición de memoria para el retorno de un número de error.

Veamos un ejemplo de programa con una llamada a la rutina ERROR:

En este capitulo hemos descrito como funcionan las rutinas del sistema operativo ISIS-II, a las que podemos acceder en un programa en ensamblador (y tambien en el len guaje PL/M). Existen ademas otras rutinas del sistema operativo y del monitor accesibles por nuestro programa, para mas detalle ver el capitulo 5 ("Uso del ISIS-II y del monitor por otros programas") del manual del usuario del ISIS-II.
En el siguiente capítulo se estudiará el uso de las rutinas en un programa de aplicación práctica, pudiéndose ver con más detalle y claridad las características y utilidades de las rutinas que se usen (OPEN, READ, WRITE, CLOSE, y ERROR).
Es un programa cuya funcion es mas que nada la aplicación de las rutinas en la practica, y ver como se disponen los parámetros en las llamas de estas rutinas.
Como una aplicacion mas práctica veremos mas tarde el programa QUILEY que se encarga de la comunicacion del sistema de desarrolo INTEL MDS-221 con el ordenador HP-3000, y del uso de la impresora y la unidad de disco del INTEL para el trasvase de datos desde y hacia el ordenador.

CAPITULO VI :

UN PROGRAMA DE APLICACION DE LA GESTION DE FICHEROS Y DEL USO DEL ISIS -II .

UN PROGRAMA DE APLICACION DE GESTION DE FICHEROS: COPIA

6.1 OBJETIVO DEL PROGRAMA

Este es un programa que va a servir como ejemplo del uso de las rutinas del sistema operativo ISIS-II en lo que a tratamiento de ficheros se refiere.El programa realiza la copia de un fichero existente en el disco (que esté en el driver) a otro fichero , creado por el programa ,con el nombre que especifiquemos.

Una vez ejecutado el programa,en el disco vamos a tener dos ficheros exactamente iguales,el fichero original (del que se hizo la copia) y el fichero creado por el programa (que es la copia del original). Ambos ficheros seran iguales en longitud y contenido,solo diferiran en el nombre.

En nuestro caso el fichero original va ser CREDIT , y el que crea el programa se llamará EDITOR;de tal manera que se podra usar el editor del sistema operativo llamando a CREDIT o bien a EDITOR.

Este programa siempre realizará la misma copia entre estos dos ficheros CREDIT y EDITOR,lo cual no es de utilidad practica pero nos sirve de base para el tratamiento de ficheros en el programa QUILEY.

6.2 DESCRIPCION DEL PROGRAMA

Primeramente tenemos las sentencias de identificacion de las rutinas,como se vió en los programas de ejemplo usando las sentencias EQU.A continuacion declaramos como externa la llamada a ISIS,esto se hace mediante la sentencia EXTRN ISIS que indica al ensamblador que la rutina ISIS no pertenece a nuestro programa,sino que será añadida posteriormente en el linkado con el SYSTEM.LIB .

La primera fase del programa esta formada por los procesos OPEN1 y OPEN2 que son los encargados de abrir los ficheros.

-OPEN1: se encarga de abrir el fichero del que vamos a hacer la copia. Como podemos ver en el bloque de parametros (OBLK1) el acceso a este fichero es solo para lectura (ACCES =1),en el caso de que en el disco con que estemos trabajando no se encuentre el fichero CREDIT se produci un error y el programa saltará a la seccion de error (ERR).

-OPEN2: se abre el fichero en el que vamos a almacenar la copia del fichero CREDIT. Si existiera ya en el disco un fichero con el nombre de EDITOR (y no esta protegido contra escritura) perderásu contenido y quedará como una copia de CREDIT una vez ejecutado nuestro programa.

Si no existe el fichero con nombre EDITOR,la rutina OPEN se encarga de crearlo y abrirlo. Como se ve en el bloque de parametros el acceso es para escritura (ACCES=2).

Cuando ya estan abiertos los dos ficheros,se pasa a la copia propiamente dicha de uno a otro y de esto se encargan los segmentos del programa READ1 y WRITE1.

-READ1: se va realizando la lectura del fichero CREDIT en bloques de 1K bytes y se almacena cada bloque en un buffer de la memoria de 1K bytes de longitud. Como se vió en la descripción de la rutina READ,existe una variable (ACTUAL) en la que tras cada operación de lectura el sistema operativo guarda el numero de bytes que se han leído realmente del fichero,de este modo aunque el contador de bytes a leer del fichero (RCNT) tenga el valor de 1024 (1K) cuando nos quede por leer el ultimo segmento de CREDIT ,que seráde longitud de menos de 1K,en la seccion de escritura (WRITE1) el contador de bytes a escribir en el fichero EDITOR tendrá el mismo valor que ACTUAL y sólo se escribiran los bytes hasta la marca de fin de fichero.

O sea,que el contador de escritura es la propia variable ACTUAL y siempre se escribira el mismo numero de bytes en EDITOR que los leídos en CREDIT .

Una vez hecha la lectura,y con el bloque en el buffer de la memoria,se pasa a la seccion de escritura.

-WRITE1: se almacena el contenido del buffer de la memoria en el fichero EDITOR. El numero de bytes que se transfieren del buffer al disco viene dado por la variable ACTUAL,como se acaba de explicar.

Tanto la seccion de lectura (READ1) como la de escritura (WRITE 1) estan dentro de un bucle del que sólo se sale cuando el valor de ACTUAL es cero, ya que esto quiere decir que hemos acabado de leer ya el último bloque del fichero CREDIT,saltándose entonces a la seccion del programa que cierra los dos ficheros (segmentos CLOSE1 y CLOSE2).

-CLOSE1: se cierra el fichero CREDIT que hemos usado como original.
-CLOSE2: se cierra el fichero EDITOR que es la copia de CREDIT.

Finalmente el programa retorna el control al sisteme operativo ISIS-II,y si hacemos una llamada al directorio de nuestro disco observaremos que hay dos ficheros con la misma longitud (CREDIT y EDITOR).

En el caso de producirse algun error debido a parametros equivocados o identificadores incorrectos en las llamadas a rutinas,el programa salta a la seccion de error (ERR) .

Este salto a la seccion de error se produce al detectar el sistema operativo un fallo cuando hacemos la llamada a alguna de sus rutinas,y lo indica en el byte de STATUS. Si el byte retorna con el valor cero quiere decir que no ha habido ningun error y la rutina se ha ejecutado normalmente,entonces nuestro programa sigue su ejecucion como estaba previsto. En el caso de que el byte STATUS no sea cero (tendrá el número de error producido) se salta a ERR, donde se hace la llamada a la rutina ERROR. Esta rutina saca por pantalla el mensaje:

ERROR nnn, USER PC mmmm

Donde nnn es el numero de error y mmmm es la direccion de retorno a nuestro programa (la lista de los números de error se encuentra en el apendice C del manual para el usuario del ISIS-II).

Con el estudio de este programa hemos visto la aplicación practica de las rutinas OPEN, READ,WRITE, CLOSE, y ERROR en un programa en ensamblador.
El uso que se hara de ellas en el programa QUILEY es similar y lo estudiaremos mas adelante.
A continuacion tenemos el listado en lenguaje ensamblador de nuestro programa COPIA.

```
ASM80 COPIA XREF PAGELENGTH(41)


ISIS-II 8080/8085 MACRO ASSEMBLER, V4.0          MODULE   PAGE    1


 LOC  OBJ           LINE        SOURCE STATEMENT

                       1 ;=================================================================
                       2 ;       Este programa realiza la copia de un fichero del disco,creando
                       3 ;       un nuevo fichero que es copia exacta del original. El fichero
                       4 ;       original es CREDIT y el generado por el programa se llamara
                       5 ;       EDITOR.Se usan las rutinas del sistema operativo OPEN, CLOSE,
                       6 ;       READ, WRITE y ERROR.
                       7 ;=================================================================
                       8 ;
                       9         CSEG
                      10 ;
 0000                 11 OPEN    EQU     0       ;Identificadores de
 0003                 12 READ    EQU     3       ;las rutinas ISIS
 0004                 13 WRITE   EQU     4
 0001                 14 CLOSE   EQU     1
 000C                 15 ERROR   EQU     12
                      16 ;
                      17         EXTRN   ISIS    ;Indicador de rutina externa (ISIS)
                      18
                      19 ;=================================================================
                      20 ;       En esta seccion se abren los dos ficheros:CREDIT y EDITOR.
                      21 ;       Si EDITOR no existe en el disco,se crea un nuevo fichero con
                      22 ;       este nombre
                      23 ;=================================================================
                      24 ;
                      25 OPEN1:                  ;Abrir fichero a leer
 0000 0E00            26         MVI     C,OPEN  ;Se carga el identificador de OPEN
 0002 111B00    C     27         LXI     D,OBLK1 ;Carga direccion bloque de parametros
 0005 CD0000    E     28         CALL    ISIS
 0008 3ACC00    C     29         LDA     STATUS  ;Carga byte STATUS
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

000B B7             30          ORA     A
000C C2C100    C    31          JNZ     ERR     ;Salto a seccion de Error si STATUS no es cero
000F 2A2500    C    32          LHLD    OAFT1   ;Se carga el AFTN del fichero abierto
0012 228300    C    33          SHLD    RAFT    ;Se guarda en RAFT
0015 22A700    C    34          SHLD    CAFT1   ;Se guarda en CAFT1
0018 C32D00    C    35          JMP     OPEN2
                    36 ;
                    37 ;
                    38 OBLK1:                   ;Bloque de parametros de OPEN
001B 2500      C    39          DW      OAFT1   ;Direccion del valor AFTN del fichero
001D 2700      C    40          DW      OFILE1  ;Direccion del nombre del fichero
001F 0100           41 ACCES1:  DW      1H      ;Control de acceso (para escritura)
0021 0000           42 ECHO1:   DW      0H      ;No hay fichero eco
0023 CC00      C    43          DW      STATUS  ;Direccion del byte STATUS
0025                44 OAFT1:   DS      2H      ;Posicion de memoria para el AFTN
0027 43524544       45 OFILE1:  DB      'CREDIT'
002B 4954
                    46 ;
                    47 ;
                    48 OPEN2:                   ;Abrir fichero a escribir
002D 0E00           49          MVI     C,OPEN  ;Carga identificador de rutina
002F 114800    C    50          LXI     D,OBLK2 ;carga direccion del bloque de parametros
0032 CD0000    E    51          CALL    ISIS
0035 3ACC00    C    52          LDA     STATUS  ;Carga byte de STATUS
0038 B7             53          ORA     A
0039 C2C100    C    54          JNZ     ERR     ;Salto a seccion de error si status no es cero
003C 2A5200    C    55          LHLD    OAFT2   ;Carga AFTN del fichero
003F 228D00    C    56          SHLD    WAFT    ;Se guarda en WAFT
0042 228D00    C    57          SHLD    CAFT2   ;Se guarda en CAFT2
0045 C35A00    C    58          JMP     READ1
```

```
LOC  OBJ           LINE        SOURCE STATEMENT

                     59 ;
                     60 ;
                     61 OBLK2:                          ;PARAMETROS OPEN2
0048 5200       C    62          DW      OAFT2         ;Direccion de OAFT2
004A 5400       C    63          DW      OFILE2        ;Direccion del nombre del fichero
004C 0200            64 ACCES:   DW      2H            ;Control de acceso (para escritura)
004E 0000            65 ECHO:    DW      0H            ;No hay fichero eco
0050 CC00       C    66          DW      STATUS        ;Direccion byte STATUS
0052                 67 OAFT2:   DS      2H
0054 45444954        68 OFILE2:  DB      'EDITOR'
0058 4F52
                     69 ;
                     70 ;=============================================================================
                     71 ;       En esta seccion del programa se lee bloque a bloque el fiche-
                     72 ;       ro CREDIT y se hace la transferencia de cada bloque a EDITOR
                     73 ;=============================================================================
                     74 ;
                     75 READ1:                          ;Lectura bloque a bloque de CREDIT
005A 0E03            76 BUCLE:   MVI     C,READ        ;Se carga el identificador
005C 118300     C    77          LXI     D,RBLK        ;Carga la direccion del bloque de parametros
005F CD0000     E    78          CALL    ISIS
0062 3ACC00     C    79          LDA     STATUS        ;Carga byte de STATUS
0065 B7              80          ORA     A
0066 C2C100     C    81          JNZ     ERR           ;Salto a error si STATUS no es cero
0069 2A9100     C    82          LHLD    ACTUAL        ;Se comprueba si el
006C 7C              83          MOV     A,H           ;numero de bytes
006D B5              84          ORA     L             ;leidos es cero
006E CA9500     C    85          JZ      CLOSE1        ;Si es cero se salta a cerrar
                     86 ;                               ;los ficheros
                     87 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    88 WRITE1:                      ;Escritura en EDITOR de los bloques leidos
0071 0E04           89          MVI     C,WRITE     ;Se carga el identificador
0073 118D00    C    90          LXI     D,WBLK      ;Se carga la direccion de parametros
0076 CD0000    E    91          CALL    ISIS
0079 3ACC00    C    92          LDA     STATUS
007C B7             93          ORA     A
007D C2C100    C    94          JNZ     ERR         ;Salto a error si STATUS no es cero
0080 C35A00    C    95          JMP     BUCLE       ;Se vuelve a leer el siguiente bloque
                    96 ;
                    97 ;
                    98 RBLK:                        ;Bloque de parametros lectura
0083                99 RAFT:    DS      2H
0085 D200      C   100          DW      BUFFER      ;Direccion del buffer de memoria
0087 0004          101 RCNT:    DW      1024        ;Contador de bytes a leer
0089 9100      C   102          DW      ACTUAL      ;Direccion de ACTUAL (bytes leidos)
008B CC00      C   103          DW      STATUS
                   104 ;
                   105 ;
                   106 WBLK:                        ;Bloque de parametros escritura
008D               107 WAFT:    DS      2H
008F D200      C   108          DW      BUFFER      ;Direccion del buffer
0091               109 ACTUAL:  DS      2           ;Numero de bytes a escribir
0093 CC00      C   110          DW      STATUS
                   111 ;
                   112 ;=============================================================================
                   113 ;       Una vez hecha la copia de un fichero a otro se pasa a cerrar
                   114 ;       ambos ficheros.
                   115 ;=============================================================================
                   116 ;
                   117 CLOSE1:                      ;Cerrar fichero leido (CREDIT)
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0095 0E01          118          MVI     C,CLOSE ;Carga identificador
0097 11A700    C   119          LXI     D,CBLK1 ;Carga direccion bloque
009A CD0000    E   120          CALL    ISIS
009D 3ACC00    C   121          LDA     STATUS
00A0 B7            122          ORA     A
00A1 C2C100    C   123          JNZ     ERR
00A4 C3AB00    C   124          JMP     CLOSE2
                   125 ;
                   126 CBLK1:                   ;PARAMETROS      CLOSE
00A7               127 CAFT1:   DS      2
00A9 CC00      C   128          DW      STATUS
                   129 ;
                   130 CLOSE2:                  ;Cerrar fichero escrito (EDITOR)
00AB 0E01          131          MVI     C,CLOSE
00AD 11BD00    C   132          LXI     D,CBLK2
00B0 CD0000    E   133          CALL    ISIS
00B3 3ACC00    C   134          LDA     STATUS
00B6 B7            135          ORA     A
00B7 C2C100    C   136          JNZ     ERR
00BA C30800        137          JMP     8H      ;Retorno al sistema operativo ISIS
                   138 ;
                   139 CBLK2:
00BD               140 CAFT2:   DS      2
00BF CC00      C   141          DW      STATUS
                   142 ;
                   143 ;=============================================================
                   144 ;               Esta es la seccion de error,a ella se salta si se
                   145 ;               produce algun error en la ejecucion de las rutinas
                   146 ;               del ISIS.
                   147 ;=============================================================
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                     148 ;
                     149 ERR:                        ;Seccion de error
  00C1 0E0C          150          MVI     C,ERROR    ;Carga identificador
  00C3 11CC00    C   151          LXI     D,EBLK     ;Carga direccion bloque
  00C6 CD0000    E   152          CALL    ISIS
  00C9 C30800        153          JMP     8H         ;Retorno al sistema operativo ISIS
                     154 ;
                     155 EBLK:                       ;Bloque de parametros de error
  00CC               156 STATUS:  DS      2H
  00CE CC00      C   157          DW      STATUS
  00D0               158          DS      2
                     159 ;
                     160 ;*************************************************************************
                     161 ;                   Zona de memoria para el buffer
                     162 ;*************************************************************************
                     163 ;
  00D2               164 BUFFER:  DS      1024       ;Buffer de memoria (de 1K bytes)
                     165          END
```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS

ACCES	C 004C	ACCES1	C 001F	ACTUAL	C 0091	BUCLE	C 005A	BUFFER	C 00D2	CAFT1	C 00A7	CAFT2	C
CBLK1	C 00A7	CBLK2	C 00BD	CLOSE	A 0001	CLOSE1	C 0095	CLOSE2	C 00AB	EBLK	C 00CC	ECHO	C
ECHO1	C 0021	ERR	C 00C1	ERROR	A 000C	ISIS	E 0000	OAFT1	C 0025	OAFT2	C 0052	OBLK1	C

```
LK2   C 0048    OFILE1 C 0027    OFILE2 C 0054    OPEN   A 0000    OPEN1  C 0000    OPEN2  C 002D    RAFT   C 008
LK    C 0083    RCNT   C 0087    READ   A 0003    READ1  C 005A    STATUS C 00CC    WAFT   C 008D    WBLK   C 008
ITE   A 0004    WRITE1 C 0071

SEMBLY COMPLETE,   NO ERRORS
```

CAPITULO VII :

UN PROGRAMA DE COMUNICACION DEL S.D. CON UN ORDENADOR Y DE GESTION DE FICHEROS.

7.1 OBJETIVOS DEL PROGRAMA

El objeto de éste programa es establecer todo el protocolo de comunicación entre un Sistema de Desarrollo Intellec MDS 221 y un Ordenador Hewlett-Packard de la serie HP-3000, así como la gestión y control de la impresora y la unidad de diskette del Sistema de Desarrollo.

Concretamente, las funciones que puede realizar son:

1. Transferencias de datos desde el teclado del Sistema de Desarrollo al ordenador.
2. Lectura de datos recibidos desde el ordenador y su transferencia a la pantalla.

Estas dos funciones, en particular, son las realizadas por cualquier terminal standard conectado a un ordenador.

3. Lectura de datos recibidos desde el ordenador y su transferencia a la impresora del Sistema de Desarrollo.
4. Lectura de datos recibidos desde el ordenador y su transferencia a la unidad de diskette del Sistema de Desarrollo.
5. Y por último, transferencia de datos desde un fichero del diskette del Sistema de Desarrollo al ordenador.

El manejo y control de todas estas funciones por parte del usuario se explicará más adelante.

El algoritmo, como puede verse en las figuras, se ha dividico en tres secciones fundamentales para un mejor entendimiento del mísmo por parte del usuario.

La explicación de éste algoritmo, así como de sus características más fundamentales se realizará exhaustivamente en el siguiente apartado.

7.2 EXPLICACION DEL ALGORITMO

En el algoritmo se pueden identificar claramente tres secciones:

- Una primera sección en la que (1) se programan los circuitos a utilizar en la comunicación, y se inicializan el puntero del buffer y los flags; y (2), se establece el bucle principal ó central de la comunicación.
- La segunda sección se encarga tanto del almacenamiento en disco del Sistema de Desarrollo, como del control de la impresora.
- La tercera sección se encarga del envío de los datos desde un fichero del disco del Sistema de Desarrollo al ordenador HP-3000.

A continuación vamos a hacer el estudio detallado de cada una de las partes de éste algoritmo.

7.2-1 SECCION DE CONTROL DE LA COMUNICACION Y DEL TECLADO

En ésta sección, el primer paso a realizar es la programación de la Usart y del Timer del Sistema de Desarrollo para el modo de transmisión específico del ordenador.

Después de esto, se resetean los flags, y se inicializa el puntero del buffer. Vamos a explicar esto más detalladamente:

Tanto en el almacenamiento en disco, como con el uso de la impresora, se utiliza un buffer de memoria en el que se almacenan los datos recibidos desde el ordenador. Es el contenido de éste buffer el que, una vez lleno, bien se almacena en disco ó se envía a la impresora dependiendo del estado áctivado ó desactivado del flag de impresora ó del de disco definidos por nosotros en el programa.

Estos flags estan almacenados en una posición de memoria a la que llamamos FLAG en nuestro programa.

Para direccionar la posición dentro del buffer en la que vamos a almacenar cada carácter leido, utilizamos una variable a la que llamamos puntero del buffer (PBUFER en nuestro programa).

Esta variable se inicializa al principio del programa con la dirección de la primera posición de memoria del buffer.

Después de inicializar el puntero del buffer entramos en el bucle central de la comunicación. En éste bucle se comprueba por un lado, si se ha presionado alguna tecla, y por otro si ha contestado el ordenador.

En caso de haberse presionado alguna tecla, se pueden dar las siguientes posibilidades:

(1) Si es la tecla HOME, se pone la Usart en la condición BREAK y se vuelve al bucle central. Esta tecla HOME realiza la función de la tecla BREAK de los terminales de la serie HP 2640/5 de Hewlett-Packard, y su misión es abortar el proceso que se está realizando en el ordenador y devolver el control al Sistema Operativo de éste.

(2) Si la tecla presionada es "Control de Impresora", se activa el flag correspondiente a la impresora, y se retorna al bucle principal.

(3) Si la tecla es "Control de Disco" se comprueba si está activado el flag de Disco. Si no lo éstá, se activa, y se abre el fichero en el que queremos almacenar los datos recibidos y volvemos al bucle principal. Si el flag ya estaba activado volvemos al bucle principal.

(4) Si es la tecla ESCAPE, se retorna al Sistema Operativo ISIS II del Sistema de Desarrollo.

(5) Si la tecla presionada es el Control de transferir fichero desde el Sistema de Desarrollo al ordenador, se abre el fichero a leer, y se salta a la tercera sección del algoritmo (bifurcación C).

(6) Si la tecla presionada no es ninguna de las anteriores, se envía a través de la Usart hacia el ordenador.

En caso de que haya contestado el ordenador, se saca por pantalla el dato recibido, y se comprueba si estan activados el flag de disco ó el de impresora. Si ninguno de los dos lo está, se vuelve al bucle principal.

Si alguno de los dos está activado, se almacena el carácter recibido en el buffer.

A continuación se comprueba si éste carácter recibido es uno de los carácteres del código ASCII D1 (11 Hexadecimal) ó D3 (13 Hex.). Estos caracteres especiales del códgo ASCII enviados por el ordenador los utilizamos para la parada autómática de la impresión y/o el almacenamiento en el disco del Sistema de Desarrollo.

En caso de que el carácter leido sea uno de estos dos, saltamos por tanto, a la sección 2 (bifurcación D) del programa.

Si no es ninguno de estos dos caracteres, incrementamos el puntero del buffer y comprobamos si éste apunta a la última posición de memoria del buffer (buffer lleno), en cuyo caso se continua en la sección 2 (bifurcación D) para imprimir ó guardar en disco el contenido de éste buffer. Si no está lleno se vuelve al bucle principal para volver a leer el siguiente carácter.

7.2-2 SECCION DE IMPRESION Y ALMACENAMIENTO EN DISCO

El primer paso a realizar en ésta sección es parar la comunicación para evitar que el ordenador continue enviando caracteres mientras se está imprimiendo o grabando en disco el buffer.

A continuación, comprobamos si está activado el flag de impresora, en cuyo caso imprimiremos el buffer carácter a carácter, bien hasta que hayamos imprimido la totalidad del contenido del buffer, ó bien hasta que detectemos un D1 ó un D3. Si se ha imprimido todo el buffer vamos a comprobar el flag de disco, y si llega un D1 ó un D3 desactivamos el flag de impresora (parada autómática), y vamos a comprobar el flag de disco. Si está activado, se lee el buffer carácter a carácter y se almacena en el disco hasta que se haya grabado la totalidad del buffer, o bien hasta que el carácter leído sea un D1 ó un D3, en cuyo caso se cierra el fichero en el que hemos estado almacenando los datos y desactivamos el flag de disco (parada automática).

Si hemos almacenado todo el buffer, ó si se ha leido un D1 ó un D3 se continua la comunicación y se salta a la bifurcación B para inicializar de nuevo el puntero del buffer y volver al bucle principal.

7.2-3 SECCION DE ENVIO DE UN FICHERO DEL SISTEMA DE DESARROLLO AL ORDENADOR

El primer paso a efectuar es la lectura dél primer bloque del fichero (que ha sido abierto para lectura) y pasarlo al buffer.

A continuación se comprueba si se ha presionado alguna tecla; en caso de que no, se comrpueba si el ordenador ha contestado.

Si se ha presionado alguna tecla, puede darse alguna de las siguientes posibilidades:

(1) Si es la tecla ESCAPE se cierra el fichero abierto para la lectura (en el sistema de Desarrollo) y se retorna al ISIS II.

(2) Si es la tecla HOME se pone la Usart en la condición BREAK, se cierra el fichero abierto para la lectura, y a continuación se salta a la bifurcación B para inicializar el puntero y volver al bucle principal.

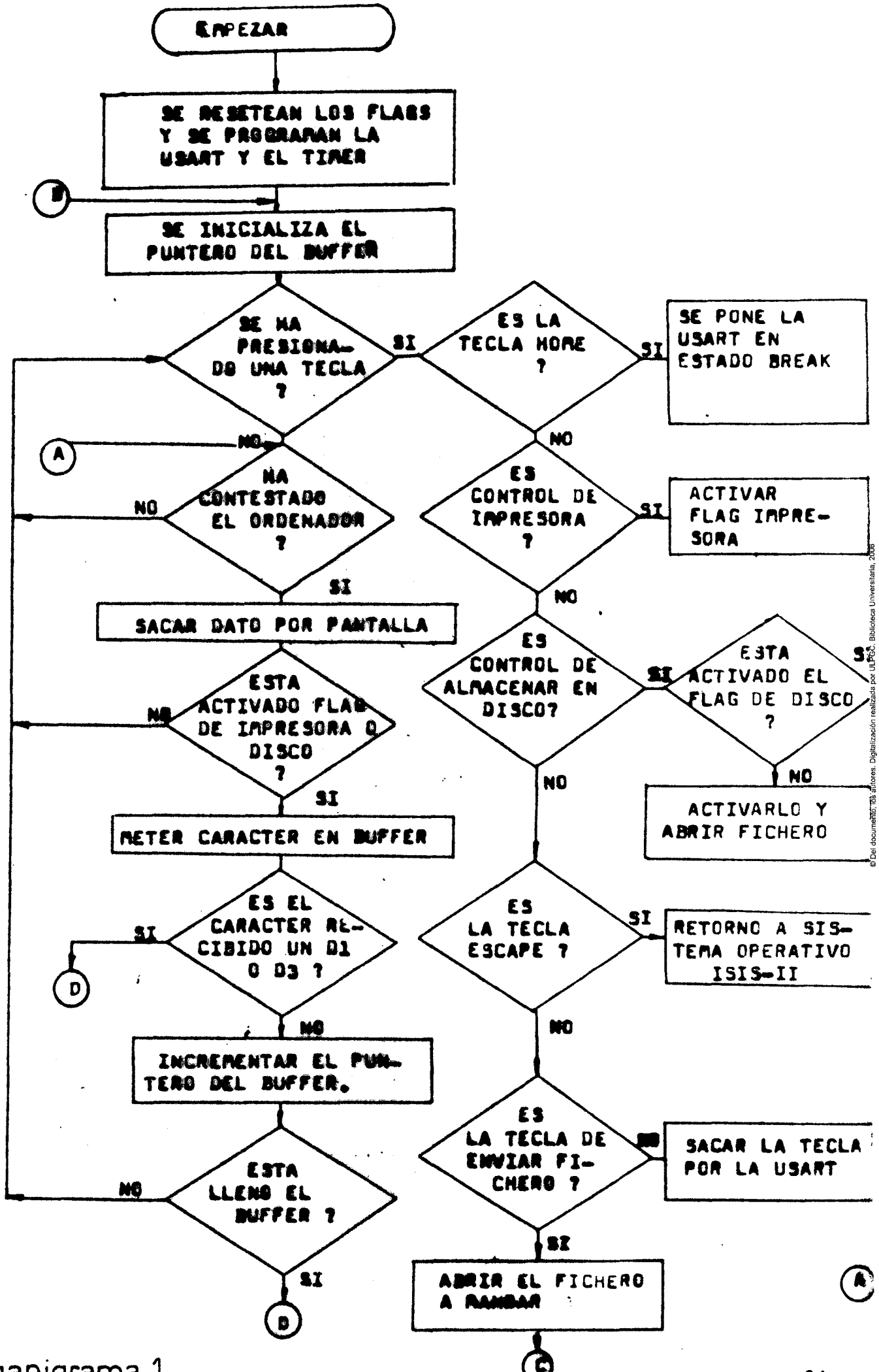
(3) Si es cualquier otra tecla, se ignora, y se vuelve,a comprobar si ha contestado el ordenador.

Si el ordenador ha contestado, sacamos el dato recibido por la Usart a la pantalla y volvemos a comprobar si se ha presionado alguna tecla.

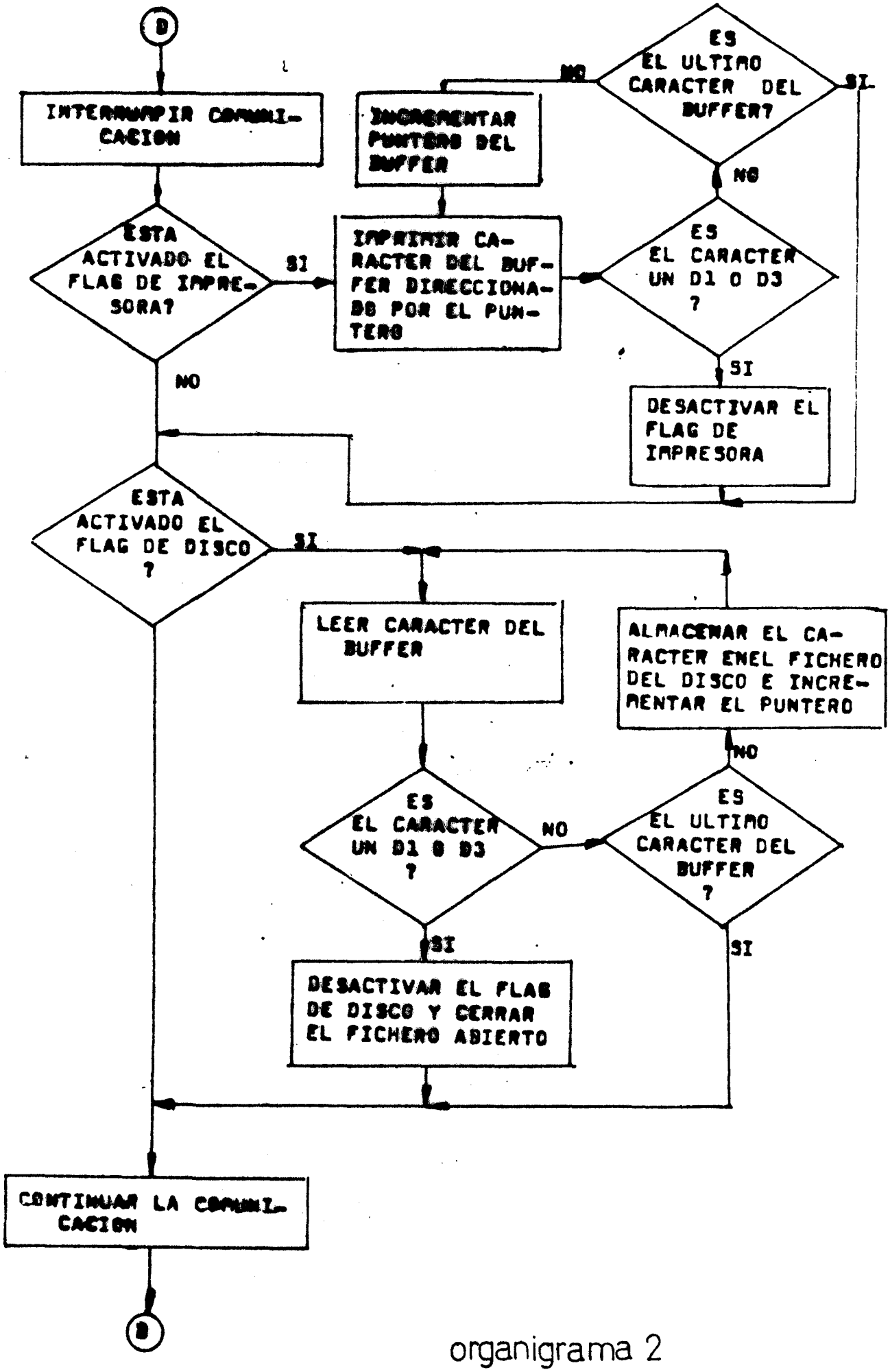
Si el ordenador no ha contestado, sw comprueba si la Usart está preparada para transmitir; si no lo está comprobamos si se ha presionado una tecla de nuevo, y si la Usart está preparada miramos si se ha leido ya el último bloque del fichero.

En caso de haber leido el último bloque del fichero, lo cerramos y se salta a la bifurcación B para inicializar el puntero y volver al bucle principal.

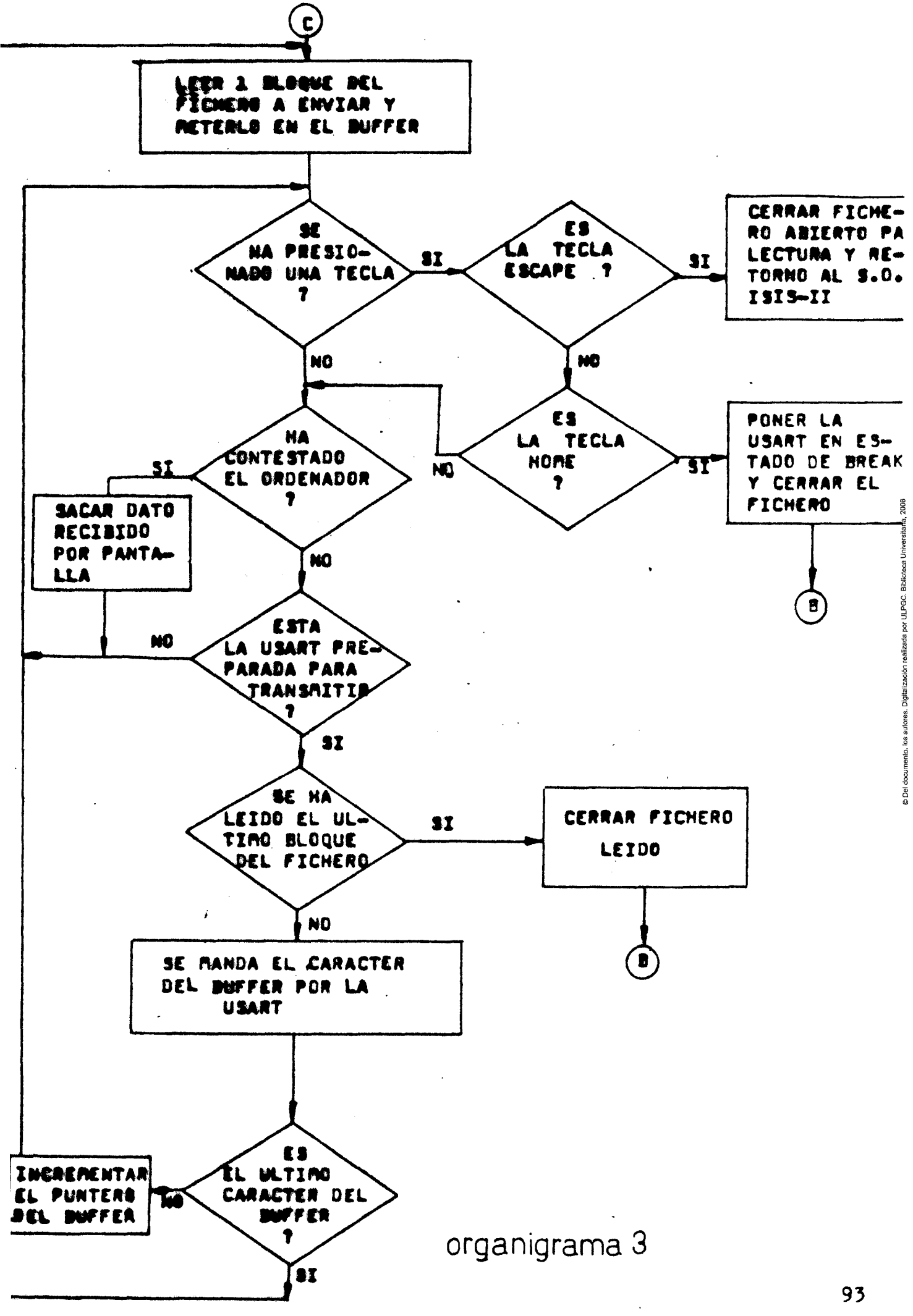
Si no se ha leido el último bloque , mandamos el carácter direccionado por el puntero del buffer al ordenador; si éste carácter es el último del buffer leemos un nuevo bloque del fichero (salto a bifurcación C); en caso de que no lo sea, incrementamos el puntero y saltamos a comprobar si se ha presionado alguna tecla de nuevo.



organigrama 1



organigrama 2



organigrama 3

7.3 DESCRIPCION DEL FUNCIONAMIENTO DEL PROGRAMA"QUILEY"

Como hemos visto anteriormente, éste programa realiza las funciones de control y gestion de la comunicación y de ficheros entre el Sistema de Desarrollo MDS 221 y el ordenador HP-3000.

Pasemos ahora a ver una descripción del funcionamiento del programa desde el punto de vista del usuario. Los pasos a seguir son:

- I) Cargar en el driver de discos del Sistema de desarrollo el disco conteniendo el programa QUILEY.
  Y bajo control del Sistema Operativo ISIS ejecutar el programa escribiendo el comando:

  - QUILEY

- II) El modem debe estar conectado al Sistema de Desarrollo; en caso contrario, en la pantalla se verá el mensaje:

  CONECTAR EL MODEM

- III) Realizar la comunicación usual con el ordenador de manera análoga a como se haría utilizando un terminal standard.

- IV) Para el uso de la impresora, proceder siguiendo los siguientes pasos:

  + a) Dar una orden de listado del programa/fichero que deseemos imprimir.

  + b) Antes de pulsar la tecla "RETURN" presionar el "CONTROL I", -de éste modo la impresora queda preparada-; en el caso de no estar encendida la impresora ó estar en OFF LINE el programa pasa a un estado de espera hasta que reciba la confirmación de "impresora preparada"
  Debe tenerse cuidado de no presionar el Control I accidentalmente sin la intención de imprimir y con la impresora apagada, puesto que el programa puede quedarse en un bucle permanente de espera.

+ c) Apretar la tecla RETURN; de éste modo el ordenador recibe la orden de listado y comienza a enviar los datos que seran imprimidos posteriormente.

+ d) La impresora se desactivará automáticamente al detectar la llegada de los caracteres D1 ó D3 (indicando fín de listado).

- V) Para almacenar en la unidad de Disco del Sistema de Desarrollo, se deben seguir los siguientes pasos:

+ a) Escribir el comando de listado del programa/fichero,a almacenar en el disco,en el ordenador.

+ b) Antes de pulsar la tecla RETURN, presionar el CONTROL K, y en pantalla aparecerá:

NOMBRE DEL FICHERO A CREAR EN EL S.D. ?

+ c) Teclear el nombre del fichero en el que queramos almacenar los datos recibidos. Si ya existe un fichero con ese nombre, pierde la información que contenía, y si no, se crea un nuevo fichero en el disco.

+ d) Pulsar la tecla RETURN. A partir de entonces el ordenador comenzará a mandar los datos a almacenar en el disco.

+ e) El disco se desactivará automáticamente al recibir uno de los dos caracteres ASCII D1 ó D3 (indicando fin de listado).

- VI) Para enviar un fichero desde el Sistema de Desarrollo al ordenador pueden seguirse varios caminos distintos:

1) - Entrar en el EDITOR (del ordenador)

- Teclear ADD

- Pulsar el CONTROL A, y aparecerá en pantalla:

NOMBRE DEL FICHERO DEL S.D. A ENVIAR ?

- Teclear el nombre del fichero teniendo en cuenta las normas del ISIS.
- Presionar la tecla RETURN.
- Una vez finalizado el envío del fichero al ordenador, utilizar el comando KEEP de éste para guardarlo en un fichero permanente del HP-3000.

2) - Entrar en el Susbsistema de copias de ficheros del ordenador: FCOPY.
(esperar hasta que el ordenador nos devuelva el signo >)
- Teclear:

FROM=;TO= (nombre del fichero);NEW

Donde:
+ (nombre del fichero) es el nombre del fichero a crear en el ordenador.
+ y NEW se utiliza para indicar que el fichero a crear es nuevo, es decir que no existe ya en un fichero permanente del ordenador.

- Pulsar el CONTROL A, y aparecerá en pantalla:

NOMBRE DEL FICHERO DEL S.D. A ENVIAR ?

- Teclear el nombre del fichero.
- Pulsar RETURN.

3) Entrar en el Subsistema BASIC.
- Pulsar el CONTROL A, y aparecerá en pantalla:

NOMBRE DEL FICHERO DEL S.D. A ENVIAR ?

- Teclear el nombre del fichero.
- Pulsar la tecla RETURN.

Una vez finalizado el envío del fichero al ordenador, utilizar el comando SAVE de éste para guardarlo en un fichero permanente del HP-3000.

Aparte de estos tres procedimientos pueden seguirse otros que gestionen el almacenamiento de información en el ordenador. Estos tres métodos son los que hemos comprobado por ahora, quedando abierta la posibilidad de usar otros subsistemas del ordenador HP-3000. Para ello basta tener en cuenta que el envío de datos desde el disco del Sist. de Desarrollo al ordenador se realiza de forma análoga al envío de de datos por teclado al ordenador.

-VII) Otras teclas y Controles de interés:

+ HOME: Pulsando ésta tecla ponemos a la Usart en la dondición BREAK, consiguiendo el mismo efecto que la tecla BREAK del terminal. (Si teníamos algún fichero abierto dentro del sistema de Desarrollo, se cerrará automáticamente).

+ ESCAPE: Pulsando ésta tecla, se interrumpe la ejecución del programa QUILEY y se devuelve el control del Sistema de Desarrollo al Sistema Operativo ISIS II.

CAPITULO VIII :

ESTUDIO DE LAS GESTIONES DE FICHEROS EN EL PROGRAMA QUILEY .

8.1 ANALISIS DE LOS PROCESOS DE TRATAMIENTO DE FICHEROS E IMPRESORA:

En lo que a procesamiento de ficheros se refire, se han usado las rutinas del sistema operativo OPEN,READ, WRITE, CLOSE y ERROR (que ya se han visto en el programa,de aplicacion practica COPIA).

Como se ha visto cuando estudiamos el funcionamiento del programa QUILEY existe un buffer de memoria (al que le hemos dado una longitud de 1024 bytes) a traves del cual se hacen las transferencias de datos del ordenador a la impresora y al disco.Mediante la sentencia :

```
TAMBUF   EQU  1024
```

se hace referencia en todo el programa a la longitud del buffer, que de este modo queda implicita con la etiqueta TAMBUF. Bastará con modificar el valor del operando de esta sentencia para darle un mayor o menor longitud al buffer según se desee.

Un vez cargado y ejecutado el programa, se produce la comunicacion del sistema de desarrollo con el oredenador HP-3000 como si de un terminal se tratara (como se describe en el proyecto "HARDWARE DEL SISTEMA DE DESARROLLO INTEL MDS-221 Y CONEXION REMOTA A UN ORDENADOR HP-3000" de Antonio Quintana Hernandez).

Para el control de la impresora y el almacenamiento en disco hemos definido un byte que se encuentra almacenado en la posicion de memoria direccionada por la etiqueta FLAG. Este byte de flags se ha definido de la siguiente forma:

Ø	Ø	Ø	Ø	UB	FD	Ø	FI

m.s.b. l.s.b.

-FI: Flag de Impresora.Este flag pasa a 1 (estado lógico) cuando se pulsa el control de impresora (CONTROL I),y pasa a Ø automaticamente cuando se cesa de imprimir.

-FD; Flag de Disco. Este flag pasa a 1 cuando se pulsa el control de disco (CONTROL K),y pasa a cero cuando se cesa de almacenar en el disco.

-UB; Flag de Ultimo Bloque. Este flag se pone a 1 cuando el programa detecta que el caracter a almacenar en el disco es un D1 o un D3, que indican que solo queda por almacenar en el disco el contenido del buffer que vandesde la primera posicion del buffer hasta que se encuentren los caracters D1 o D3. Este flag se pone a Ø cuando se ha finalizado de almacenar en disco.

Inicialmente los flags se encuentran todos a Ø (esto se hace en la inicialización de los flags, antes de establecer la comunicacion con el ordenador). Los flags de impresora o de disco pasaran a 1 cuando pulsemos los controles de imp. o disco respectivamente.
Cuando en el bucle principal de la comunicacion (BUCPRI) se detecta que se ha presionado una tecla, y esta tecla es control de impresora (CONTROL I) se produce una bifurcacion en el programa a CFLAGI,donde se pone a 1 el flag de impresora y se retorna al bucle BUCPRI para reanudar la comunicacion con el ordenador y almacenar los datos recibidos a continuacion en el buffer de la memoria.
En el caso de que la tecla pulsada sea control de almacenamiento en disco (CONTROL K) el programa salta a la bifurcacion CFLAGD, donde se pone a 1 el flag de disco y se retorna a la comunicacion para almacenar los datos en el buffer. Ademas de esto se abre un fichero en el disco (el fichero en que vamos a almacenar los datos rec ibidos), esto se hace en la seccion del programa a partir de la etiqueta ABFICH. Primeramente se saca por pantalla el mensaje:

NOMBRE DEL FICHERO A CREAR EN EL S.D. ?

Para sacar este mensaje se hace uso de la subrutina SDDPP, que saca por pantalla un comentario cuya direccion se carga en el par de registros HL. Cuando se ha tecleado el nombre del fichero (se detecta el final del nombre cuando la tecla apretada es "RETURN") se procede a abrirlo usando la rutina OPEN tal como ya se ha explicado en el ejemplo de aplicacion practica (nuestro programa COPIA), la diferencia aquíestriba en que el nombre del ficherom a abrir es el que ha especificado el usuario. Una vez abierto el fichero se reanuda la comunicacion para ir almacenando los datos recibidos en el buffer.

Tanto el almacenamiento en disco como el uso de la impresora se hacen seccionando los datos recibidos desde el ordenador en bloques de 1024 bytes, que es la longitud del buffer de memoria que usamos en el programa.

Los datos recibidos se van almacenando en el buffer hasta que se llene (han llegado 1024 bytes) o bien hasta que se reciban los caracteres D1 o D3 (Indican que el ordenador ha cesado de enviar datos),de esta funcion se encarga el segmento de programa a partir de la etiqueta SPUSAR. Cuando se ha acabdo de almacenar los datos en el buffer (por una de las dos razones que acabamos de explicar),el programa salta a la bifurcacion RDIOGD,donde primeramente se interrumpe la comunicacion con el ordenador para que cese de enviar datos y luego se comprueba si está activado el flag de impresora o de disco.

Si el flag activado es el de impresora el programa salta a la bifurcacion IMPRIM, mediante la cual se va imprimiendo - caracter a caracter - el contenido del buffer de memoria hasta que se imprima en su totalidad, o bien hasta que se detecten los caracteres D1 o D3. En el caso de que se halla imprimido todo el contenido del buffer (el puntero del buffer apunta al último caracter) se reanuda la comunicacion para recibir mas datos , colocarlos en el buffer y luego imprimirlos. Si se ha detectado uno de los carcteres D1 o D3 se cesa de imprimir, se desactiva el flag de impresora (FI=0) y se vuelve al bucle principal de c omunicacion (BUCPRI). Para ver con detalle las secciones de interrumpir y reanudar la comunicacion ver el proyecto "HARDWARE DEL SISTEMA DE DESARROLLO INTEL MDS-221 Y CONEXION REMOTA A UN ORDENADOR HP-3000".
El caracter (byte) a imprimir pasa del buffer de memoria al registro C y se envía a la impresora a traves del port F8, en la seccion de programa de uso de la impresora hay unos bucles (LAZO1, LAZO2, LAZO3 y LAZO4) que comprueban el estado de la impresora, de este modo si la impresora no esta preparada (encendida, en ON LINE y READY) el programa se queda bloqueado en estos bucles y para salir de ellos habrá que resetear el sistema, cuando se hace uso de la impresora el programa enmascara todas las interrupciones del usuario. Se debe tener la precaucion de no pulsar el control de Impresora accidentalmente y con la impresora apagada,pues entonces veremos como el programa queda bloqueado y habra que resetear el sistema.

Si el flag activado es el flag de disco (FD=1) el programa salta a la bifurcacion ALDISC, donde se pasa el contenido de buffer al disco. Para ello se hace uso de la rutina WRITE como ya se ha visto en el programa de ejemplo COPIA, en este caso hay una diferencia y es que el numero de bytes a almacenar en el fichero del disco (variable CONESC) se obtiene

leyendo previamente el contenido del buffer, pudiendo ocurrir que:
-En el bloque de datos de 1024 bytes almacenado en el buffer no halla ni un D1 ni un D3, entonces se almacenan los 1024 bytes del buffer, y la varable CONLEC tiene el valor Ø4ØØ (1024 en hexadecimal).
-Que se detecten los caracteres D1 o D3, entonces el valor de CONLEC es el numero de bytes que va desde la primera posicion del buffer hasta la posicion del caracter D1 o D3 dentro del buffer.
Todo este proceso se realiza en la seccion de programa que va desde la etiqueta OCHO hasta la NUEVE.
En el caso de que en el buffer se hayan detectado un D1 o un D3 se activa el flag de Ultimo Bloque (UB=1) para indicar, una vez hecha la transferncia al disco, que se debe saltar a cerrar el fichero.
El proceso de cerrar el fichero se hace en el segmento de programa a partir de la etiqueta CEFICH, se usa la rutina CLOSE tal como se ha visto en el programa de aplicación . Cuando se ha cerrado el fichero se salta a DIEZ donde se ponen a Ø los flags de disco y ultimo bloque y se retorna a RDIOGD para continuar la comunicacion con el ordenador y se salta al bucle principal BUCPRI.

Hemos visto hasta ahora las partes del programa que conciernen con el tratamiento de los datos que nos envía el ordenador, ahora veremos la seccion que trata del envío de un fichero del disco del sistema de desarrollo al ordenador HP-3000.
Si se pulsa el control de enviar fichero (CONTROL A) el programa salta del bucle principal BUCPRI a la bifurcacion ENVFIC, donde primero se abre el fichero que vamos a leer usando la misma seccion de programa que abre el fichero para almacenar. La diferencia es que aquí el acceso es para lectura y el comentario que sale por pantalla es:

NOMBRE DEL FICHERO DEL S.D, A LEER?

Cuando se ha abierto el fichero, se manda un RETURN automaticamente al ordenador para que ejecute el comando que habiamos tecleado antes de pulsar el contrl A (ver descripcion del funcionamiento del programa), y se pasa a LECFIC donde se va leyendo el fichero en bloques de 1024 bytes (de forma analoga a como vimos el uso de la rutina READ en el programa COPIA). Como se ve en el bloque de parámetros (RBLK) el contador de bytes a leer del fichero es de 1024 bytes (RCNT=1024), y el numero de bytes realmente leídos del fichero lo indica la variable CONLEC.
Hecha la operacion de lectura (pasar los bytes del disco al buffer) se

comprueba si se ha pulsado alguna tecla, por medio del bucle LAZPRI (de estructura similar al bucle de comunicacion BUCPRI), en caso de haberse pulsado alguna tecla se salta a la bifurcacion SDT donde se comprueba si la tecla es ESCAPE o HOME. Si no es ninguna de estas dos se ignora y el programa vuelve a LAZPRI, ahora bien, en el caso de que sea:

-ESCAPE: se cierra el fichero que se ha abierto para lectura y se retorna el control al sistema operativo.

-HOME: se envía al ordenador la señal de BREAK (interrupcion del proceso que se estaba ejecutando) y se cierra el fichero abierto para lectura.

El uso de estas dos teclas durante el envío de un fichero del disco al ordenador se creyó conveniente para poder interrumpir la transferencia de un modo practico en caso de haber algun error.

Si no hemos pulsado ninguna tecla (o bien cualquier otra distinta de HOME o ESCAPE) lo primero que hace el programa es comprobar si la variable CONLEC es cero,si es cero quiere decir que en la operacion de lectura previa (LECFIC anterior) no se han leído ningun byte del fichero,o sea que ya hemos leidom todo el fichero (cuando se ha leído todo el fichero el marcador interno de lectura apunta al final,y ya no se leen mas bytes) como sucedía en el programa COPIA. Si CONLEC no es cero significa que se ha leído un numero de bytes dado, y se envían al ordenador tantos caracteres del buffer como indique el valor de CONLEC.

Una vez enviado el contenido del buffer al ordenador se va a leer el siguiente bloque del fichero del disco (salto a LECFIC), si el bloque leido era el ultimo el valor de CONLEC despues,de la nueva lectura será cero,y el programa salta a la bifurcacion EXIT. En esta seccion se cierra el fichero enviado y se manda automaticamente un CONTROL Y para indicar al ordenador que se ha finalizado el envío de datos ,luego se salta a la etiqueta INDBUF donde se inicializa el puntero del buffer y se vuelve al bucle de comunicacion BUCPRI.

Debido a la alta velocidada con que se en enviaban los datos del disco al ordenador se producian gran cantidad de errores (perdida de caracteres y líneas del fichero, e insercion de caracteres extraños en el fichero creado por el ordenador) . Para evitar estos errores se disminuyóla velocidad de envio de caracteres insertando una una subrutina de retraso (DELAY) despues del envío de cada caracter ASCII LF (Line Feed) para dar tiempo a que se asimilen los datos en viados antes de crear una nueva línea.

En la subrutina DELAY se efectua un bucle decrementando un contador formado por el par de registros DE, al tiempo que se sacan por pantalla los datos que puedan llegar del ordenador durante ese tiempo.

Como se hizo en el programa COPIA, cuando se produce un error enlas llamadas a las rutinas del sistema operativo se interrumpe la ejecucion del programa y se vuelve al control del sistema operativo. Esto sucederácuando, por ejemplo, tecleemos un nombre que no este de acuerdo con el formato de nombre de ficheros del sistema operativo cuando respondamos a la pregunta:

NOMBRE DEL FICHERO DEL S.D. A LEER ?

veremos entonces como cesa la comunicacion con el ordenador y sale en pantalla el mensaje que nos indica que estamos bajo el control del sistema operativo ISIS-II.

El estudio que se ha realizado del programa QUILEY en este apartado ha sido un estudio a nivel de bloques agrupando las distintas partes del programa segun la funcion que cumplían. A continuacion tenemos el listado de las instrucciones en ensamblador, que con ayuda delos comentarios añadidos y las etiqutas podran identificarse y analizarse para tener un estudio a nivel de ensamblador de nuestro programa.

```
ASM80 QUILE6.SRC XREF PAGELENGTH(41)

ISIS-II 8080/8085 MACRO ASSEMBLER, V4.0          MODULE    PAGE     1

 LOC  OBJ          LINE        SOURCE STATEMENT

                     1 ;*************************************************************************
                     2 ;   Este programa mantiene la comunicacion entre un Sistema de Desarrollo In-
                     3 ;  tellec MDS 221 y un Ordenador de la serie HP-3000. La comunicacion es:
                     4 ;  Serie, Asincrona *64, velocidad de transmision 1200 baudios, 8 bits de lon-
                     5 ;  gitud del caracter, sin paridad, y con 2 bits de stop.
                     6 ;
                     7 ;   Asimismo, realiza la gestion y control de la impresora, y de la Unidad de
                     8 ;  diskette del Sistema de Desarrollo. Para ello, se utilizan los siguientes
                     9 ;  controles:
                    10 ;
                    11 ;              - CONTROL I: Esta tecla se utiliza para el manejo de la impre-
                    12 ;                  sora. Para imprimir, basta con pulsar "CONTROL I" a partir
                    13 ;                  del momento en que queramos comenzar a imprimir. La parada
                    14 ;                  es automatica cuando se termina el listado.
                    15 ;
                    16 ;              - CONTROL K: Esta tecla se utiliza para almacenar en el flopy
                    17 ;                  del Sistema de Desarrollo cualquier programa o fichero del
                    18 ;                  del ordenador. Para que realice esta funcion basta con pul-
                    19 ;                  sar CONTROL I a partir del momento en que queramos almace-
                    20 ;                  nar, si bien es recomendable que se haga despues de escri-
                    21 ;                  bir el comando de listado del ordenador y antes de pulsar la
                    22 ;                  tecla RETURN.
                    23 ;
                    24 ;              - CONTROL A: Esta tecla se utiliza para enviar un fichero desde
                    25 ;                  la unidad de flopy del Sistema de Desarrollo al ordenador.
                    26 ;                  Para ello basta con entrar en el Subsistema del Ordenador
                    27 ;                  en que queramos volcar el fichero, y cuando el ordenador
                    28 ;                  este preparado esperando los datos, pulsar CONTROL A.
                    29 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    30 ;                - HOME: Esta tecla se utiliza para enviar un condicion BREAK
                    31 ;                    al ordenador.
                    32 ;
                    33 ;                - ESCAPE: Esta tecla se utiliza para abortar la ejecucion
                    34 ;                    de este programa y devolver el control al Sistema Opera-
                    35 ;                    tivo ISIS II del Sistema de Desarrollo.
                    36 ;
                    37 ;*************************************************************************
                    38 ;
                    39          EXTRN   CI,CO,CSTS,ISIS
                    40          CSEG
                    41 ;
000D                42 DISABL   EQU     0DH
0005                43 ENABL    EQU     05H
00FF                44 CPUC     EQU     0FFH
00F8                45 PIOI     EQU     0F8H
00F8                46 PIOO     EQU     0F8H
00F9                47 PIOS     EQU     0F9H
00F9                48 PIOC     EQU     0F9H
0004                49 FO       EQU     04H
0002                50 IBF      EQU     02H
0001                51 OBF      EQU     01H
0001                52 DEVRDY   EQU     01H
0014                53 LPTC     EQU     014H
0015                54 LSTC     EQU     015H
0011                55 D1       EQU     11H     ; Caracteres que controlan la parada automatica
0013                56 D3       EQU     13H
0009                57 TCIMP    EQU     09H     ; Ctl I: Tecla para activar la impresora
000B                58 TCDISK   EQU     0BH     ; Ctl K: Tecla para crear un fichero en el flopy
001B                59 TRISIS   EQU     1BH     ; ESCAPE: Tecla de Retorno a Isis
```

```
LOC  OBJ           LINE        SOURCE STATEMENT

007F                 60 RBOUT   EQU     7FH
0008                 61 BSPACE  EQU     08H
0013                 62 CPCOMU  EQU     13H     ; Ctl S: control para parar la comunicacion
0011                 63 CCCOMU  EQU     11H     ; Ctl Q: control para continuar la comunicacion
0000                 64 OPEN    EQU     0       ; Parametros para control de la Rutina Isis
0003                 65 READ    EQU     3
0004                 66 WRITE   EQU     4
0001                 67 CLOSE   EQU     1
000C                 68 ERROR   EQU     12
0010                 69 BAUD    EQU     010H    ; Velocidad de Transmision
0400                 70 TAMBUF  EQU     1024    ; Longitud ,en bytes, del buffer
0001                 71 TADISK  EQU     01H     ; Ctl A: Tecla para enviar un fichero desde el flopy
0019                 72 MFF     EQU     019H    ; Ctl Y: Marca de Fin de fichero
001D                 73 HOME    EQU     1DH     ; HOME: Tecla para enviar la condicion BREAK.
                     74 ;
                     75 ;
                     76 PRINCP:
0000 218904    C     77         LXI     H,FLAG  ; Reseteo de los flags
0003 3600            78         MVI     M,0H
0005 3E40            79         MVI     A,40H   ; Se resetea la Usart
0007 D3F7            80         OUT     0F7H
0009 3ECF            81         MVI     A,0CFH  ; Programacion de la Usart del Canal 2
000B D3F7            82         OUT     0F7H
                     83 ;
000D 3E76            84         MVI     A,076H  ;  Se programa el Timer para trabajar
000F D3F3            85         OUT     0F3H    ; en modo 3 con una frecuencia de
0011 211000          86         LXI     H,BAUD  ; 76.8 Khz.
0014 7D              87         MOV     A,L
0015 D3F1            88         OUT     0F1H
0017 7C              89         MOV     A,H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0018 D3F1           90          OUT     0F1H
                    91 ;
001A 3E07           92          MVI     A,07H   ; Se activan las lineas DTR y los flags
001C D3F7           93          OUT     0F7H    ; R*E y T*E de la Usart.
                    94 ;
001E 21C203   C     95          LXI     H,INICIA
0021 CD7303   C     96          CALL    SDDPP
                    97 ;
0024 DBF7           98          IN      0F7H    ; Se comprueba si esta activada la linea
0026 E680           99          ANI     80H     ; DSR del modem.
0028 C23800   C    100          JNZ     UNO
002B 218C03   C    101          LXI     H,CEM
002E CD7303   C    102          CALL    SDDPP
0031 DBF7          103 DOS:     IN      0F7H    ; Si no esta activada, se espera a que lo este
0033 E680          104          ANI     80H     ; despues de sacar el comentario por pantalla.
0035 CA3100   C    105          JZ      DOS
                   106 ;
0038 3E37          107 UNO:     MVI     A,37H   ; Se esta activada, se activan entonces: DTR,
003A D3F7          108          OUT     0F7H    ; RTS, R*E, y T*E.
                   109 ;
003C 218C04   C    110 INDBUF:  LXI     H,BUFFER; Inicializacion del puntero del buffer
003F 228A04   C    111          SHLD    PBUFER
                   112 ;
0042 CD0000   E    113 BUCPRI:  CALL    CSTS    ; Bucle principal:
0045 0F            114          RRC
0046 DC5300   C    115          CC      KEYB    ;   - Si se pulsado una tecla salta a KEYB
0049 DBF7          116 RESPUE:  IN      0F7H
004B E602          117          ANI     02H
004D C22701   C    118          JNZ     SPUSAR  ;   - Si el ordenador ha contestado salta a SPUSAR
0050 C34200   C    119          JMP     BUCPRI
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   120 ;
                   121 ;
0053 CD0000    E   122 KEYB:   CALL    CI        ; Se lee la tecla pulsada.
0056 4F            123         MOV     C,A
0057 FE1D          124         CPI     HOME      ; Si no es la tecla HOME salta a KEYB1
0059 C26B00    C   125         JNZ     KEYB1
005C 3E2F          126         MVI     A,2FH     ; Si es la tecla HOME se pone a la Usart
005E D3F7          127         OUT     0F7H      ; en estado BREAK momentaneamente
0060 CD2403    C   128         CALL    DELAY
0063 3E37          129         MVI     A,37H
0065 D3F7          130         OUT     0F7H
0067 CD5802    C   131         CALL    CEFICH
006A C9            132         RET
006B FE09          133 KEYB1:  CPI     TCIMP     ; Si la tecla pulsada es Ctl I salta a
006D CA9100    C   134         JZ      CFLAGI    ; CFLAGI
0070 FE0B          135         CPI     TCDISK    ; Si es Ctl K salta a CFLAGD
0072 CA9C00    C   136         JZ      CFLAGD
0075 FE1B          137         CPI     TRISIS    ; Si es ESCAPE retorna al ISIS.
0077 CA0800        138         JZ      08H
007A FE01          139         CPI     TADISK    ; Si es Ctl A salta a ENVFIC.
007C CA7902    C   140         JZ      ENVFIC
007F FE7F          141         CPI     RBOUT     ; Si es Rubout, sustituye su codigo por
0081 C28600    C   142         JNZ     TRES
0084 0E08          143         MVI     C,BSPACE; el de Backspace
                   144 ;
                   145 ;*****************************************************************
                   146 ;
                   147 ;   Esta Rutina envia el caracter que hay en el registro C
                   148 ;   hacia la Usart.
                   149 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0086 DBF7          150 TRES:   IN      0F7H    ; - Se espera a que la Usart este preparada
0088 E601          151         ANI     01H     ; para transmitir.
008A CA8600    C   152         JZ      TRES
008D 79            153         MOV     A,C     ; - Salida del caracter que hay en el registro
008E D3F6          154         OUT     0F6H    ; C por la Usart
0090 C9            155         RET
                   156 ;
                   157 ;*****************************************************************
                   158 ;
                   159 ;
                   160 ;*****************************************************************
                   161 ;
                   162 ;    Rutina de activacion del flag de impresora
                   163 ;
0091 3A8904    C   164 CFLAGI: LDA     FLAG    ; - Si se ha pulsado Ctl I, se activa
0094 F601          165         ORI     01H
0096 328904    C   166         STA     FLAG
0099 C34900    C   167         JMP     RESPUE
                   168 ;
                   169 ;*****************************************************************
                   170 ;
                   171 ;
                   172 ;*****************************************************************
                   173 ;
                   174 ;  Esta rutina activa el flag de disco y abre un nuevo fichero
                   175 ;
009C 3A8904    C   176 CFLAGD: LDA     FLAG    ; - Si se ha pulsado Ctl K se comprueba si es-
009F E604          177         ANI     04H     ; ta activado el flag de disco.
00A1 C24900    C   178         JNZ     RESPUE  ; - Si lo esta, se retorna
00A4 F604          179         ORI     04H     ; - Y si no lo esta, se activa, y se abre el
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00A6 328904   C    180          STA     FLAG     ;nuevo fichero
00A9 210200        181          LXI     H,02H    ; - Parametro para que el nuevo fichero sea
00AC 220F01   C    182          SHLD    ACCES    ; abierto para escritura
00AF 21A203   C    183          LXI     H,NDF
                   184 ;*****************************************************************
                   185 ;
                   186 ;       Rutina para abrir un fichero, estando el comentario
                   187 ;       a sacar en pantalla preguntando por el nombre del
                   188 ;       programa en los registros HL cuando se hace la llamada
                   189 ;
                   190 ;*****************************************************************
                   191 ;
00B2 CD7303   C    192 ABFICH: CALL    SDDPP
00B5 0610          193          MVI     B,10H
00B7 211701   C    194          LXI     H,NFILE
00BA CD0000   E    195 CINCO:  CALL    CSTS     ; - Se lee el nombre del fichero a abrir, y
00BD 0F            196          RRC              ; se almacena en las posiciones de memoria
00BE D2BA00   C    197          JNC     CINCO    ; a partir de NFILE
00C1 CD0000   E    198          CALL    CI
00C4 4F            199          MOV     C,A
00C5 FE7F          200          CPI     RBOUT
00C7 C2D400   C    201          JNZ     CIN
00CA 2B            202          DCX     H
00CB 04            203          INR     B
00CC 0E08          204          MVI     C,08H
00CE CD0000   E    205          CALL    CO
00D1 C3BA00   C    206          JMP     CINCO
00D4 CD0000   E    207 CIN:    CALL    CO
00D7 79            208          MOV     A,C
00D8 FE0D          209          CPI     0DH
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00DA CAE300    C   210          JZ      SEIS
00DD 05            211          DCR     B
00DE 77            212          MOV     M,A
00DF 23            213          INX     H
00E0 C3BA00    C   214          JMP     CINCO
00E3 3620          215 SEIS:    MVI     M,20H
00E5 23            216          INX     H
00E6 05            217          DCR     B
00E7 C2E300    C   218          JNZ     SEIS
                   219 ;
                   220 ;
00EA 0E0A          221          MVI     C,0AH   ; Se saca un <LF> automatico por pantalla
00EC CD0000    E   222          CALL    CO
                   223 ;
00EF 0E00          224          MVI     C,OPEN
00F1 110B01    C   225          LXI     D,OBLK
00F4 CD0000    E   226          CALL    ISIS
00F7 3A2501    C   227          LDA     STATUS
00FA B7            228          ORA     A
00FB C27F03    C   229          JNZ     ERR
00FE 2A1501    C   230          LHLD    OAFT
0101 229F02    C   231          SHLD    RAFT
0104 225002    C   232          SHLD    WAFT
0107 226A02    C   233          SHLD    CAFT
010A C9            234          RET
                   235 ;
010B 1501      C   236 OBLK:    DW      OAFT
010D 1701      C   237          DW      NFILE
010F               238 ACCES:   DS      2H
0111 0000          239 ECO:     DW      0H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0113 2501      C   240          DW      STATUS
0115               241 OAFT:    DS      2H
0117               242 NFILE:   DS      14
0125               243 STATUS:  DS      2H
                   244 ;
                   245 ;
                   246 ;
                   247 ;*****************************************************************
                   248 ;
                   249 ;    Rutina para leer los datos que llegan desde el ordenador
                   250 ;    y sacarlos por pantalla. En caso de que haya activado al-
                   251 ;    gun flag, se almacena el dato en el BUFFER.
                   252 ;
                   253 ;*****************************************************************
0127 DBF6          254 SPUSAR:  IN      0F6H     ; - Se lee el dato
0129 E67F          255          ANI     7FH      ; - Se resetea el octavo bit
012B 4F            256          MOV     C,A
012C FE05          257          CPI     05H      ; - Si ha llegado un Enquiry (protocolo) se
012E CA6201    C   258          JZ      CUATRO   ; salta a cuatro
0131 CD0000    E   259          CALL    CO
0134 3A8904    C   260          LDA     FLAG     ; - Se saca el dato por pantalla, y si no hay
0137 E605          261          ANI     05H      ; ningun flag activado se vuelve al bucle
0139 CA4200    C   262          JZ      BUCPRI   ; principal.
013C 2A8A04    C   263          LHLD    PBUFER   ; - Si hay activado algun flag se mete el dato
013F 71            264          MOV     M,C      ; en el buffer
0140 79            265          MOV     A,C
0141 FE11          266          CPI     D1       ; - Si el dato es un D1 o un D3 (parada automa-
0143 CA6A01    C   267          JZ      RDIOGD   ;tica), se salta a RDIOGD.
0146 FE13          268          CPI     D3
0148 CA6A01    C   269          JZ      RDIOGD
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

014B 23           270         INX     H       ; - Se incrementa el puntero del buffer
014C 228A04    C  271         SHLD    PBUFER
014F 118B08    C  272         LXI     D,BUFFER+TAMBUF-1
0152 2A8A04    C  273         LHLD    PBUFER
0155 7B           274         MOV     A,E
0156 BD           275         CMP     L       ; - Si el buffer esta lleno se salta a
0157 C24200    C  276         JNZ     BUCPRI  ; RDIOGD, si no se vuelve al bucle
015A 7A           277         MOV     A,D     ; principal.
015B BC           278         CMP     H
015C C24200    C  279         JNZ     BUCPRI
015F C36A01    C  280         JMP     RDIOGD
0162 0E06         281 CUATRO: MVI     C,06H   ; - Si llego un Enquiry se contesta automati-
0164 CD8600    C  282         CALL    TRES    ; camente con un Acknowledge
0167 C34200    C  283         JMP     BUCPRI
                  284 ;
                  285 ;
016A 0E13         286 RDIOGD: MVI     C,CPCOMU; Se para la comunicacion
016C CD8600    C  287         CALL    TRES
016F DBF7         288 HOLA:   IN      0F7H    ; - Se lee el ultimo caracter que llego
0171 E602         289         ANI     02H     ; a la Usart cuando se paro la comunica-
0173 CA6F01    C  290         JZ      HOLA    ; cacion, y se almacena en el buffer.
0176 DBF6         291         IN      0F6H
0178 E67F         292         ANI     7FH
017A 4F           293         MOV     C,A
017B CD0000    E  294         CALL    CO
017E 2A8A04    C  295         LHLD    PBUFER
0181 71           296         MOV     M,C
0182 3A8904    C  297         LDA     FLAG    ; - Se leen los flags.
0185 E601         298         ANI     01H     ; - Si esta activado el flag de impresora
0187 C49A01    C  299         CNZ     IMPRIM  ; se salta a IMPRIM
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

018A 3A8904   C   300          LDA     FLAG
018D E604         301          ANI     04H      ; - Si esta activado el flag de disco
018F C40402   C   302          CNZ     ALDISC   ; se salta a ALDISC
0192 0E11         303          MVI     C,CCCOMU; - Se continua la comunicacion y vuelve
0194 CD8600   C   304          CALL    TRES     ; a inicializar el puntero del buffer.
0197 C33C00   C   305          JMP     INDBUF   ; y al bucle principal
                  306 ;
                  307 ;
                  308 ;*************************************************************************
                  309 ;
                  310 ;    Rutina para imprimir el contenido del buffer
                  311 ;
                  312 ;*************************************************************************
                  313 ;
019A 118C04   C   314 IMPRIM: LXI     D,BUFFER;
019D EB           315 SIETE:  XCHG              ; - Se carga en HL la direccion de comienzo del buffer
019E 4E           316         MOV     C,M       ; - Se lee el caracter y se almacena en el registro C
019F 1615         317         MVI     D,LSTC    ; - Se cargan los comandos de control de la impresora
01A1 1E14         318         MVI     E,LPTC
01A3 CDCD01   C   319         CALL    OUTDVR
01A6 79           320         MOV     A,C
01A7 FE11         321         CPI     D1
01A9 CAC201   C   322         JZ      SALIDA
01AC FE13         323         CPI     D3
01AE CAC201   C   324         JZ      SALIDA
01B1 23           325         INX     H
01B2 EB           326         XCHG
01B3 2A8A04   C   327         LHLD    PBUFER
01B6 23           328         INX     H
01B7 7B           329         MOV     A,E
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

01B8 BD            330          CMP     L
01B9 C29D01    C   331          JNZ     SIETE
01BC 7A            332          MOV     A,D
01BD BC            333          CMP     H
01BE C29D01    C   334          JNZ     SIETE
01C1 C9            335          RET
                   336 ;
01C2 3A8904    C   337 SALIDA:  LDA     FLAG
01C5 E6FC          338          ANI     0FCH
01C7 E6FC          339          ANI     0FCH
01C9 328904    C   340          STA     FLAG
01CC C9            341          RET
                   342 ;
01CD 3E0D          343 OUTDVR:  MVI     A,DISABL
01CF D3FF          344          OUT     CPUC
01D1 DBF9          345 LAZO1:   IN      PIOS
01D3 E607          346          ANI     F0 OR IBF OR OBF
01D5 C2D101    C   347          JNZ     LAZO1
01D8 7A            348          MOV     A,D
01D9 D3F9          349          OUT     PIOC
01DB DBF9          350 LAZO2:   IN      PIOS
01DD E607          351          ANI     F0 OR IBF OR OBF
01DF FE01          352          CPI     OBF
01E1 C2DB01    C   353          JNZ     LAZO2
01E4 DBF8          354          IN      PIOI
01E6 E601          355          ANI     DEVRDY
01E8 CAD101    C   356          JZ      LAZO1
01EB DBF9          357 LAZO3:   IN      PIOS
01ED E607          358          ANI     F0 OR IBF OR OBF
01EF C2EB01    C   359          JNZ     LAZO3
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

01F2 7B            360          MOV     A,E
01F3 D3F9          361          OUT     PIOC
01F5 DBF9          362 LAZO4:   IN      PIOS
01F7 E607          363          ANI     FO OR IBF OR OBF
01F9 C2F501    C   364          JNZ     LAZO4
01FC 79            365          MOV     A,C
01FD D3F8          366          OUT     PIOO
01FF 3E05          367          MVI     A,ENABL
0201 D3FF          368          OUT     CPUC
0203 C9            369          RET
                   370 ;
                   371 ;*************************************************************************
                   372 ;
                   373 ;
                   374 ;
                   375 ;*************************************************************************
                   376 ;
                   377 ;       Rutina para contar el numero de bytes a almacenar en el disco
                   378 ;
                   379 ;*************************************************************************
0204 110000        380 ALDISC: LXI     D,0H     ; - Se inicializa en DE el contador de bytes
0207 218C04    C   381          LXI     H,BUFFER; - Se carga en HL la direccion de comienzo del buffer
020A 7E            382 OCHO:    MOV     A,M      ; - Se lee el caracter
020B FE11          383          CPI     D1
020D CA1502    C   384          JZ      VEINTE   ; - Si es el caracter de parada automatica salta a
0210 FE13          385          CPI     D3       ;   VEINTE
0212 C22002    C   386          JNZ     TREINT
0215 3A8904    C   387 VEINTE:  LDA     FLAG     ; - Se activa el flag de "Ultimo bloque de disco"
0218 F608          388          ORI     08H
021A 328904    C   389          STA     FLAG
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

021D C33202    C   390          JMP     NUEVE
0220 13            391 TREINT:  INX     D        ; - Se incrementa el contador de bytes y se compara
0221 010004        392          LXI     B,TAMBUF;   con la longitud del buffer. Si son iguales
0224 79            393          MOV     A,C      ;   salta a nueve
0225 B8            394          CMP     E
0226 C22E02    C   395          JNZ     CINCUE
0229 78            396          MOV     A,B
022A BA            397          CMP     D
022B CA3202    C   398          JZ      NUEVE
022E 23            399 CINCUE:  INX     H        ; - Se incrementa HL para leer el siguiente
022F C30A02    C   400          JMP     OCHO     ;   caracter
0232 EB            401 NUEVE:   XCHG             ;  - Se almacena en CONESC el numero de bytes
0233 225402    C   402          SHLD    CONESC   ;    a guardar en el disco
                   403 ;
                   404 ;
                   405 ;*************************************************************************
                   406 ;
                   407 ;      Rutina para escribir en el fichero abierto definido por su WAFT
                   408 ;       el numero de bytes especificado por CONESC
                   409 ;
                   410 ;*************************************************************************
                   411 ;
0236 0E04          412 ESCRIT:  MVI     C,WRITE
0238 115002    C   413          LXI     D,WBLK
023B CD0000    E   414          CALL    ISIS
023E 3A2501    C   415          LDA     STATUS
0241 B7            416          ORA     A
0242 C27F03    C   417          JNZ     ERR
0245 3A8904    C   418          LDA     FLAG
0248 E608          419          ANI     08H
```

```
LOC  OBJ           LINE         SOURCE STATEMENT

024A E608            420          ANI     08H
024C C25802    C     421          JNZ     CEFICH
024F C9              422          RET
                     423 ;
                     424 WBLK:
0250                 425 WAFT:    DS      2H
0252 8C04      C     426          DW      BUFFER
0254                 427 CONESC:  DS      2H
0256 2501      C     428          DW      STATUS
                     429 ;
                     430 ;
                     431 ;*****************************************************************************
                     432 ;
                     433 ;       Rutina para cerrar el fichero definido por su CAFT
                     434 ;
                     435 ;*****************************************************************************
0258 0E01            436 CEFICH:  MVI     C,CLOSE
025A 116A02    C     437          LXI     D,CBLK
025D CD0000    E     438          CALL    ISIS
0260 3A2501    C     439          LDA     STATUS
0263 B7              440          ORA     A
0264 C27F03    C     441          JNZ     ERR
0267 C36E02    C     442          JMP     DIEZ
                     443 ;
                     444 CBLK:
026A                 445 CAFT:    DS      2H
026C 2501      C     446          DW      STATUS
026E 3A8904    C     447 DIEZ:    LDA     FLAG
0271 E6F3            448          ANI     0F3H
0273 E6F3            449          ANI     0F3H
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0275 328904   C    450          STA     FLAG
 0278 C9            451          RET
                    452 ;
                    453 ;
                    454 ;*****************************************************************
                    455 ;
                    456 ;      Rutina para enviar un fichero desde el Sistema de Desarrollo
                    457 ;       al Ordenador
                    458 ;
                    459 ;*****************************************************************
                    460 ENVFIC:
 0279 210300        461          LXI     H,03H    ; - Parametro para abrir el fichero para solo
 027C 220F01   C    462          SHLD    ACCES    ; lectura
 027F 216A04   C    463          LXI     H,NDFAE
 0282 CDB200   C    464          CALL    ABFICH   ; - Se abre el fichero
 0285 0E0D          465          MVI     C,0DH    ; - Se manda <CR> automatico a la Usart
 0287 CD8600   C    466          CALL    TRES
 028A CD2403   C    467          CALL    DELAY
                    468 ;
                    469 ;*****************************************************************
                    470 ;
                    471 ;          Rutina para leer y volcar en el buffer el fichero
                    472 ;          definido por su RAFT, almacenandose el numero de bytes
                    473 ;          leidos en CONLEC
                    474 ;
                    475 ;*****************************************************************
 028D 0E03          476 LECFIC: MVI     C,READ
 028F 119F02   C    477          LXI     D,RBLK
 0292 CD0000   E    478          CALL    ISIS
 0295 3A2501   C    479          LDA     STATUS
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0298 B7            480          ORA     A
0299 C27F03    C   481          JNZ     ERR
029C C3AB02    C   482          JMP     CUAREN
                   483 ;
                   484 RBLK:
029F               485 RAFT:    DS      2
02A1 8C04      C   486          DW      BUFFER
02A3 0004          487 RCNT:    DW      TAMBUF  ; Se especifica el numero de bytes a leer
02A5 A902      C   488          DW      CONLEC
02A7 2501      C   489          DW      STATUS
02A9               490 CONLEC:  DS      2       ; La rutina Isis almacena aqui el numero de bytes
                   491 ;                          leidos realmente
                   492 ;
                   493 ;*******************************************************************************
                   494 ;
                   495 ;       Rutina para enviar todo el contenido del buffer al ordenador
                   496 ;
                   497 ;*******************************************************************************
02AB 210000        498 CUAREN:  LXI     H,0H    ; Se inicializa el puntero del buffer
02AE 228A04    C   499          SHLD    PBUFER
02B1 CD0000    E   500 LAZPRI:  CALL    CSTS    ; Lazo principal:
02B4 0F            501          RRC              ;    - Si se ha pulsado una tecla salta a SDT
02B5 DA4F03    C   502          JC      SDT
02B8 DBF7          503          IN      0F7H    ;    - Si el ordenador ha contestado salta a
02BA E602          504          ANI     02H     ;      LECDAT
02BC C20103    C   505          JNZ     LECDAT
02BF DBF7          506          IN      0F7H    ;    - Y si la Usart no esta preparada para
02C1 E601          507          ANI     01H     ;      transmitir se vuelve al lazo principal
02C3 CAB102    C   508          JZ      LAZPRI
02C6 2AA902    C   509          LHLD    CONLEC  ; Si el contador del numero real de bytes leidos
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

02C9 7C            510          MOV     A,H      ; es cero se salta a EXIT
02CA B5            511          ORA     L
02CB CA1903    C   512          JZ      EXIT
02CE 2A8A04    C   513          LHLD    PBUFER   ; El contenido del puntero del buffer (PBUFER) se
02D1 EB            514          XCHG             ; suma a la direccion de comienzo del buffer y se
02D2 218C04    C   515          LXI     H,BUFFER; almacena en los registros HL
02D5 7D            516          MOV     A,L
02D6 83            517          ADD     E
02D7 6F            518          MOV     L,A
02D8 7C            519          MOV     A,H
02D9 8A            520          ADC     D
02DA 67            521          MOV     H,A
02DB 4E            522          MOV     C,M      ; - Se lee un caracter del buffer y se saca
02DC CD8600    C   523          CALL    TRES     ; por la Usart
02DF 79            524          MOV     A,C
02E0 FE0A          525          CPI     0AH      ; - Si el caracter enviado es un LF se salta
02E2 CC2403    C   526          CZ      DELAY    ; a DELAY
02E5 2AA902    C   527          LHLD    CONLEC   ; - Si el puntero del buffer (PBUFER) no es
02E8 2B            528          DCX     H        ; igual al numero real de bytes leidos (CONLEC)
02E9 7B            529          MOV     A,E      ; se salta a SIGBLO para continuar con el siguien-
02EA BD            530          CMP     L        ; te caracter.
02EB C2F902    C   531          JNZ     SIGCAR
02EE 7A            532          MOV     A,D
02EF BC            533          CMP     H
02F0 C2F902    C   534          JNZ     SIGCAR
02F3 CD2403    C   535          CALL    DELAY    ; - Si PBUFER=CONLEC se efectua un retraso para
02F6 C38D02    C   536          JMP     LECFIC   ; para volver a leer el siguiente bloque.
02F9 EB            537 SIGCAR:  XCHG
02FA 23            538          INX     H        ; - Se incrementa el puntero del buffer.
02FB 228A04    C   539          SHLD    PBUFER
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 02FE C3B102   C    540          JMP     LAZPRI
                    541 ;
 0301 DBF6          542 LECDAT:  IN      0F6H     ; - Se lee el dato enviado por el ordenador
 0303 E67F          543          ANI     7FH
 0305 4F            544          MOV     C,A
 0306 FE05          545          CPI     05H      ; - Si es un Enquiry se salta a PROTOC
 0308 CA1103   C    546          JZ      PROTOC
 0308 CD0000   E    547          CALL    CO       ; - Si no, se saca el dato por pantalla y se
 030E C3B102   C    548          JMP     LAZPRI   ; vuelve al lazo principal
 0311 0E06          549 PROTOC:  MVI     C,06H    ; - Si llego un Enquiry se contesta con un Ack-
 0313 CD8600   C    550          CALL    TRES     ; nowledge y se vuelve al lazo principal
 0316 C3B102   C    551          JMP     LAZPRI
                    552 ;
 0319 CD5802   C    553 EXIT:    CALL    CEFICH   ; - Se cierra el fichero leido, y se envia al
 031C 0E19          554          MVI     C,MFF    ; ordenador una marca de "Fin de Fichero" (Ctl Y).
 031E CD8600   C    555          CALL    TRES     ; Se vuelve a inicializar el puntero del buffer y al
 0321 C33C00   C    556          JMP     INDBUF   ; bucle principal.
                    557 ;
                    558 ;
                    559 ;*************************************************************************
                    560 ;
                    561 ;         Rutina para efectuar un retraso especificado por el valor del
                    562 ;         par de registros DE, y que a su vez lee todos los datos que
                    563 ;         puedan llegar desde el ordenador durante este tiempo
                    564 ;
                    565 ;*************************************************************************
                    566 ;
 0324 D5            567 DELAY:   PUSH    D
 0325 11FF2F        568          LXI     D,2FFFH
 0328 DBF7          569 DEL1:    IN      0F7H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

032A E602           570         ANI     02H
032C CA3F03    C    571         JZ      DEL2
032F DBF6           572         IN      0F6H
0331 E67F           573         ANI     7FH
0333 FE05           574         CPI     05H
0335 CA4703    C    575         JZ      DEL3
0338 4F             576         MOV     C,A
0339 CD0000    E    577         CALL    CO
033C C32803    C    578         JMP     DEL1
033F 1B             579 DEL2:   DCX     D
0340 7B             580         MOV     A,E
0341 B2             581         ORA     D
0342 C22803    C    582         JNZ     DEL1
0345 D1             583         POP     D
0346 C9             584         RET
0347 0E06           585 DEL3:   MVI     C,06H
0349 CD8600    C    586         CALL    TRES
034C C32803    C    587         JMP     DEL1
                    588 ;
                    589 ;
                    590 ;
034F CD0000    E    591 SDT:    CALL    CI      ; Esta rutina lee la tecla pulsada cuando se estaban
0352 FE1B           592         CPI     TRISIS  ; enviando datos desde un fichero al ordenador (solo
0354 C25D03    C    593         JNZ     SDT1    ; lee las teclas ESCAPE y HOME).
0357 CD5802    C    594         CALL    CEFICH
035A C30800         595         JMP     08H     ;    - Si la tecla pulsada es ESCAPE, cierra el fichero
035D FE1D           596 SDT1:   CPI     HOME    ;      abierto y vuelve al ISIS.
035F C2B102    C    597         JNZ     LAZPRI
0362 3E2F           598         MVI     A,2FH   ;    - Si es HOME, pone a la Usart en estado de BREAK,
0364 D3F7           599         OUT     0F7H    ;      cierra el fichero abierto, y vuelve a iniciali-
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0366 CD2403   C    600          CALL     DELAY    ;       zar el puntero del buffer y al bucle principal.
 0369 3E37          601          MVI      A,37H
 036B D3F7          602          OUT      0F7H     ;     - Si es cualquier otra tecla, la ignora y continua
 036D CD5802   C    603          CALL     CEFICH   ;       la ejecucion normal del programa.
 0370 C33C00   C    604          JMP      INDBUF
                    605 ;
                    606 ;
                    607 ;*******************************************************************************
                    608 ;
                    609 ;       Rutina para presentar en pantalla cualquier cadena de caracteres,
                    610 ;       cuya direccion de comienzo este especificada en el par de registros
                    611 ;       HL, y cuyo ultimo caracter sea: "&"
                    612 ;
                    613 ;*******************************************************************************
                    614 ;
 0373 7E           615 SDDPP:   MOV      A,M
 0374 FE26         616          CPI      '&'
 0376 C8           617          RZ
 0377 4F           618          MOV      C,A
 0378 CD0000   E    619          CALL     CO
 037B 23           620          INX      H
 037C C37303   C    621          JMP      SDDPP
                    622 ;
                    623 ;
                    624 ;*******************************************************************************
                    625 ;
                    626 ;      Rutina para presentar en pantalla los errores de Isis cometidos en
                    627 ;      la gestion de un fichero del disco.
                    628 ;
                    629 ;*******************************************************************************
```

```
 LOC  OBJ           LINE         SOURCE STATEMENT

 037F 0E0C            630 ERR:    MVI     C,ERROR
 0381 118803    C     631         LXI     D,EBLK
 0384 CD0000    E     632         CALL    ISIS
 0387 C9              633         RET
                      634 ;
 0388 2501      C     635 EBLK:   DW      STATUS
 038A                 636         DS      2H
                      637 ;
                      638 ;
                      639 ;******************** COMENTARIOS A PRESENTAR EN PANTALLA **********************
                      640 ;
                      641 ;
 038C 0D              642 CEM:    DB      0DH,0AH,'CONECTAR EL MODEM',0DH,0AH,'&'
 038D 0A
 038E 434F4E45
 0392 43544152
 0396 20454C20
 039A 4D4F4445
 039E 4D
 039F 0D
 03A0 0A
 03A1 26
 03A2 0D              643 NDF:    DB      0DH,0AH,'NOMBRE DEL FICHERO A CREAR ? &'
 03A3 0A
 03A4 4E4F4D42
 03A8 52452044
 03AC 454C2046
 03B0 49434845
 03B4 524F2041
 03B8 20435245
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

03BC 4152203F
03C0 2026
03C2 1B            644 INICIA: DB      1BH,48H,1BH,4AH,'***************** DESARROLLADO POR ANTONIO '
03C3 48
03C4 1B
03C5 4A
03C6 2A2A2A2A
03CA 2A2A2A2A
03CE 2A2A2A2A
03D2 2A2A2A2A
03D6 20444553
03DA 4152524F
03DE 4C4C4144
03E2 4F20504F
03E6 5220414E
03EA 544F4E49
03EE 4F20
03F0 5155494E      645         DB      'QUINTANA Y CARLOS LEY *****************',0DH
03F4 54414E41
03F8 20592043
03FC 41524C4F
0400 53204C45
0404 59202A2A
0408 2A2A2A2A
040C 2A2A2A2A
0410 2A2A2A2A
0414 2A2A
0416 0D
0417 2A2A2A2A      646         DB      '************************ SISTEMA DE'
041B 2A2A2A2A
```

```
 LOC  OBJ          LINE         SOURCE STATEMENT

 041F 2A2A2A2A
 0423 2A2A2A2A
 0427 2A2A2A2A
 042B 2A2A2A20
 042F 53495354
 0433 454D4120
 0437 4445
 0439 20444553      647          DB       ' DESARROLLO PREPARADO **************************',0DH,0AH,'&'
 043D 4152524F
 0441 4C4C4F20
 0445 50524550
 0449 41524144
 044D 4F202A2A
 0451 2A2A2A2A
 0455 2A2A2A2A
 0459 2A2A2A2A
 045D 2A2A2A2A
 0461 2A2A2A2A
 0465 2A2A
 0467 0D
 0468 0A
 0469 26
 046A 0D            648 NDFAE:   DB       0DH,0AH,'NOMBRE DEL FICHERO A LEER ? &'
 046B 0A
 046C 4E4F4D42
 0470 52452044
 0474 454C2046
 0478 49434845
 047C 524F2041
 0480 204C4545
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  0484 52203F20
  0488 26
                     649 ;
                     650 ;
                     651 ;********************************* ZONA DE MEMORIA *******************************
                     652 ;
  0489               653 FLAG:   DS      1
  048A               654 PBUFER: DS      02H
  048C               655 BUFFER: DS      TAMBUF
                     656 ;
  0000            C  657         END     PRINCP


PUBLIC SYMBOLS


EXTERNAL SYMBOLS
CI      E 0000     CO      E 0000     CSTS    E 0000     ISIS    E 0000

USER SYMBOLS
ABFICH  C 00B2     ACCES   C 010F     ALDISC  C 0204     BAUD    A 0010     BSPACE  A 0008     BUCPRI  C 0042
CAFT    C 026A     CBLK    C 026A     CCCOMU  A 0011     CEFICH  C 0258     CEM     C 038C     CFLAGD  C 009C
CI      E 0000     CIN     C 00D4     CINCO   C 00BA     CINCUE  C 022E     CLOSE   A 0001     CO      E 0000
CONLEC  C 02A9     CPCOMU  A 0013     CPUC    A 00FF     CSTS    E 0000     CUAREN  C 02AB     CUATRO  C 0162
D3      A 0013     DEL1    C 0328     DEL2    C 033F     DEL3    C 0347     DELAY   C 0324     DEVRDY  A 0001
DISABL  A 000D     DOS     C 0031     EBLK    C 0388     ECO     C 0111     ENABL   A 0005     ENVFIC  C 0279
ERROR   A 000C     ESCRIT  C 0236     EXIT    C 0319     FLAG    C 0489     FO      A 0004     HOLA    C 016F
IBF     A 0002     IMPRIM  C 019A     INDBUF  C 003C     INICIA  C 03C2     ISIS    E 0000     KEYB    C 0053
LAZO1   C 01D1     LAZO2   C 01DB     LAZO3   C 01EB     LAZO4   C 01F5     LAZPRI  C 02B1     LECDAT  C 0301
LPTC    A 0014     LSTC    A 0015     MFF     A 0019     NDF     C 03A2     NDFAE   C 046A     NFILE   C 0117

BUFFER  C 048C
CFLAGI  C 0091
CONESC  C 0254
D1      A 0011
DIEZ    C 026E
ERR     C 037F
HOME    A 001D
KEYB1   C 006B
LECFIC  C 028D
NUEVE   C 0232
```

```
OAFT    C 0115    OBF     A 0001    OBLK    C 010B    OCHO    C 020A    OPEN    A 0000    OUTDVR  C 01CD
PIOC    A 00F9    PIOI    A 00F8    PIOO    A 00F8    PIOS    A 00F9    PRINCP  C 0000    PROTOC  C 0311
RBLK    C 029F    RBOUT   A 007F    RCNT    C 02A3    RDIOGD  C 016A    READ    A 0003    RESPUE  C 0049
SDDPP   C 0373    SDT     C 034F    SDT1    C 035D    SEIS    C 00E3    SIETE   C 019D    SIGCAR  C 02F9
STATUS  C 0125    TADISK  A 0001    TAMBUF  A 0400    TCDISK  A 000B    TCIMP   A 0009    TREINT  C 0220
TRISIS  A 001B    UNO     C 0038    VEINTE  C 0215    WAFT    C 0250    WBLK    C 0250    WRITE   A 0004

ASSEMBLY COMPLETE,    NO ERRORS
```

```
PBUFER  C 048A
RAFT    C 029F
SALIDA  C 01C2
SPUSAR  C 0127
TRES    C 0086
```

BIBLIOGRAFIA

-INFORMATION SYSTEM CONCEPTS FOR MANAGEMENT. De Henry C.Lucas, Jr.
(Editorial McGraw-Hill)

-OPERATING SYSTEMS. De Madnick/Donovan.
(Editorial McGraw-Hill)

-INTRODUCTION TO COMPUTER SYSTEMS. de Glenn H.Mac Ewen.
(Editorial McGraw-Hill)

-VAX TECHNICAL SUMMARY. De la casa DIGITAL.

-ISIS-II USER'S GUIDE. Del INTEL MICROCOMPUTER DEVELOPEMENT SYSTEM.