CENTRO DE INVESTIGACION CIENTIFICA Y
DE EDUCACION SUPERIOR DE ENSENADA

DESARROLLO DE UN SISTEMA DE MULTI-TAREAS EN TIEMPO REAL CON UN AMBIENTE DE VENTANAS PARA LA IBM-PC

T E S I S

MAESTRIA EN CIENCIAS

David Octavio Villalobos Fierro

ENSENADA, BAJA CALIFORNIA, FEBRERO DE 1991.

CENTRO DE INVESTIGACION CIENTIFICA Y DE EDUCACION
SUPERIOR DE ENSENADA
(CICESE)

CENTRO DE COMPUTO

DESARROLLO DE UN SISTEMA DE MULTITAREAS EN TIEMPO REAL CON UN AMBIENTE DE VENTANAS PARA LA IBM-PC

TESIS

Que para cubrir parcialmente los requisitos necesarios para obtener el grado de **MAESTRO EN CIENCIAS** presenta:

DAVID OCTAVIO VILLALOBOS FIERRO.

Febrero 1991

Ensenada, Baja California. México.

RESUMEN de la Tesis de David Octavio Villalobos Fierro presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en CIENCIAS COMPUTACIONALES. Ensenada, B. C., México. Febrero de 1991.

DESARROLLO DE UN SISTEMA DE MULTITAREAS EN TIEMPO REAL CON UN AMBIENTE DE VENTANAS PARA LA IBM-PC.

Resumen aprobado por:

M. en C. Hugo Homero Hidalgo Silva
Director de Tesis.

Multitareas es una técnica que puede ser implantada en una computadora (con un solo procesador) para proporcionarle la capacidad de lograr el efecto de ejecutar varios programas a la vez, de manera que parezca ante el usuario que los programas están corriendo simultáneamente.

El diseño y desarrollo consistió en un sistema de multitareas en tiempo real con un ambiente de ventanas para la IBM-PC o computadoras 100% compatibles con IBM, para ejecutarse sobre la cima del sistema operativo MS-DOS.

El recurso clave que se administra es la unidad de procesamiento central (UPC) bajo el **esquema de prevaciado.** Se implantó el **calendarizado "Round-robin"** con la finalidad de compartir equitativamente la atención de la UPC entre los procesos.

A cada proceso se le asigna una **tabla de control** que contiene información referente al proceso, necesaria para que pueda ejecutarse eficientemente en el sistema de multitareas y en el ambiente de ventanas.

Cada proceso tiene **su propia pila** de trabajo privada y tiene asignada **una ventana.** A cada proceso se le es asignado **un nodo o celda** en una cola implantada en una lista ligada circular, que constituye su tabla de control.

Para lograr la señalización en **tiempo real,** se interceptó la interrupción **08H** del sistema básico de entradas-salidas (BIOS). Esta señal llama a un **módulo despachador** de tareas que es el encargado de realizar la conmutación de los procesos, cada vez que le llegue una señal del temporizador a una tasa de **18.2 veces por segundo.** Entonces realiza el **salvaguardado de contexto y ambiente** del proceso interrumpido y obtiene ésta misma información del proceso por ejecutar de la tabla de control correspondiente para cada proceso.

El mecanismo que se utiliza para controlar el procesamiento de multitareas es mediante los **accesos al sistema** que proporciona tanto el BIOS así como el sistema operativo MS-DOS.

Los vectores de interrupción del **BIOS** que se **interceptaron,** además de la INT 08H, son: La INT 09H para controlar el teclado y propocionar una interacción con el usuario mediante un procesador de mandos básicos. La INT 10H para controlar el video y proporcionar un ambiente de manejo de ventanas. La INT 1BH, para conmutar entre los procesos activos y el procesador de mandos y la INT 0BH para establecer comunicación a través del puerto serie.

Los vectores de interrupción del **MS-DOS** que se interceptaron son: La INT 21H para controlar las principales funciones que proporciona el sistema operativo y poder hacer que sea reentrante. La INT 24H para el manejador de errores críticos. Se **crearon** un grupo de interrupciones: La INT 82H, 83H y 84H para el buen desempeño del MS-DOS en un ambiente reentrante.

TESIS DEFENDIDA POR: **DAVID OCTAVIO VILLALOBOS FIERRO**

Y APROBADA POR EL SIGUIENTE COMITE:

M.C. HUGO HOMERO HIDALGO SILVA. - Director del Comité

M.C. FERNANDO FAVELA VARA. - Miembro del Comité

M.C. JOSE LUIS BRISEÑO CERVANTES. - Miembro del Comité

M.C. ALEJANDRO HINOJOSA CORONA.- Miembro del Comité

DR. SERGIO REYES COCA.- Miembro del Comité

ING. SALVADOR CASTAÑEDA AVILA.- Jefe del Centro de Computo

DR. HECTOR ECHAVARRIA HERAS.- Director Académico

5 DE MARZO DE 1991

cicese

DEDICATORIA

DIOS me dá la oportunidad por segunda ocación de dedicar en forma expresa mi agradecimiento y cariño a mi querida familia Villalobos Fierro.

A mi madre Rebeca Fierro de Villalobos, de quien sigo obteniendo ejemplo de esfuerzo y valentía para enfrentarme a la vida. Reiterandole nuevamente mi admiración y cariño.

A mis hermanos Javier Reynaldo, Jaime Joel, Carlos Edgardo, Rebeca Lorena y Luis Manuel, por su hermandad y compañerísmo, de quienes siento un apoyo contínuo ante todas las metas que me fijo en la vida.

A mi padre Luis Villalobos Guerrero, de quien guardo un hermoso recuerdo.

A mis familiares.

A mi novia Armida Hernández González y Haiitie Hereen Hernández González por su cariño y apoyo moral que me han proporcionado durante estos últimos años, llenandome de felicidad y tranquilidad con su comprensión y compañía.

AGRADECIMIENTOS

A mi Director de Tesis, M. en C. Hugo Homero Hidalgo Silva, por su acertada dirección y comentarios contínuos.

Al Ing. Salvador Castañeda, Jefe del Centro de Cómputo del CICESE, por su contínuo apoyo y beca proporcionada durante la etapa final del trabajo de tesis.

Al CONACyT por su beca otorgada para los estudios de Maestría, que ahora finalizo.

A todos los miembros del Comité de Tesis: M. en C. Fernando Favela Vara, M. en C. José Luis Briseño Cervantes, M. en C. Alejandro Hinojosa Corona y al Dr. Sergio Reyes Coca, por sus comentarios y sugerencias.

A Moisés Cywiack y Francisco Lazo Valencia, por su disponibilidad a transmitir sus conocimientos, comentarios y sugerencias durante el trabajo en grupo del desarrollo del Núcleo del Sistema de Multitareas.

A Laura Martínez Ríos Del Río por su disponibilidad y desinteresada ayuda en el asesoramiento de los procesadores de texto.

A Concepción Mendoza por la atención y revisión prestada al texto y contenido de la tesis.

A todos mis compañeros, amigos y maestros que de alguna u otra forma me transmitieron sus conocimientos y sugerencias para obtener el grado que actualmente intento alcanzar.

CONTENIDO

Página

I.- INTRODUCCION. 1

II.- CARACTERISTICAS DE PROGRAMACION PRINCIPALES DE LA IBM-PC. 3

II.1.- Mapa de memoria de la IBM-PC (XT). 3

II.2.- Sistema operativo. 6

II.2.1.- Sistema básico de entradas y salidas (BIOS). 7

II.2.2.- Sistema operativo MS-DOS. 11

II.3.- Accesos a los servicios del sistema. 17

II.3.1.- Manejo de interrupciones. 17

II.3.2.- Llamadas al BIOS. 18

II.3.3.- Llamadas al sistema operativo MS-DOS. 21

III.- TEORIA DE SISTEMAS OPERATIVOS. 22

III.1.- Procesos. 23

III.2.- Estados de los procesos. 24

III.3.- Calendarizado de la unidad de procesamiento central (UPC). 26

III.4.- Manejo de memoria. 30

III.5.- Condiciones de carreras. 31

III.6.- Secciones críticas. 32

III.7.- Semáforos. 32

III.8.- Abrazo mortal. 34

III.9.- Código reentrante. 35

III.10.- Comunicación interprocesos. 36

III.11.- Sistemas de tiempo real. 38

III.12.- Multitareas. 38

IV.- COMUNICACION EN SERIE ASINCRONA. 40

V.- DISEÑO DEL SISTEMA DE MULTITAREAS. 42

V.1.- Estructuras de datos. 43

CONTENIDO (Continuación)

Página

V.2.- Núcleo del sistema de multitareas. 47
V.3.- Interfaz usuario (ventanas). 55
V.4.- Procesador de mandos. 57
VI.- CONCLUSIONES Y RECOMENDACIONES. 60
APENDICE 64
BIBLIOGRAFIA 72

LISTA DE FIGURAS.

Figura		Página
1	Distribución de la memoria convencional de la IBM.	4
2	Distribución de la memoria convencional al inicializar el sistema.	9
2.1	Distribución de la memoria convencional con las tablas de interrupción y las tablas del ROM BIOS.	10
2.2	Distribución de la memoria convencional cuando es cargado el sector del cargador del sistema operativo.	11
2.3	Una posible configuración del bloque de parámetros del BIOS (BPB).	11
2.4	Distribución de la memoria convencional cuando han sido cargados los archivos del sistema IO.SYS y MSDOS.SYS	12
2.5	Distribución de la memoria convencional cuando el cargador SYSINIT y el archivo MSDOS.SYS han sido reubicados.	14
2.6	Distribución de la memoria convencional cuando han sido cargadas las tablas de asignación de archivos del MS-DOS, área de memoria reservada a disco, bloques de control de archivos y controladores de dispositivos instalables.	15
2.7	Distribución de la memoria convencional cuando ha sido cargado el procesador de mandos COMMAND.COM.	16

LISTA DE FIGURAS (Continuación).

Figura		Página
3	(a) Multiprogramación de cuatro procesos. (b) Modelo conceptual de cuatro procesos secuenciales independientes. (c) Solamente un programa esta utilizando la UPC en un instante dado.	24
4	Diagrama básico que muestra los estados por los que puede pasar un proceso. Mostrando las transiciones o funciones entre los estados.	25
5	Multitareas, conmutando de un proceso a otro haciendo transparente las implantaciones.	40
6	Formato de dato via trasmisión serie.	41

LISTA DE TABLAS.

Tabla		Página
I	Características de los tres principales tipos de sistemas operativos.	7
II	Vectores de interrupción para el microprocesador 8088/8086.	19
III	Vectores de interrupción para el controlador de interrupciones 8259-A.	19
IV	Puntos de entrada al BIOS.	20
V	Interrupciones proporcionadas al usuario.	20
VI	Apuntadores a las tablas de datos del BIOS.	20
VII	Interrupciones proporcionadas por el MS-DOS.	22
VIII	Vectores de interrupción que se interceptaron y redirigieron.	48
IX	Nuevos vectores de interrupción.	48

I.- INTRODUCCION.

Las nuevas posibilidades que proporciona la tecnología actual a nivel internacional, mediante el uso de las computadoras personales en la adaptación y desarrollo de sistemas, han provocado una revolución en el desarrollo de programas. Se ha facilitado la interacción hombre-máquina, mediante adecuadas interfaces-usuario obteniendo una mayor optimización en el desarrollo de actividades asistidas por computadoras.

Multitareas es una técnica para lograr que la computadora realice el efecto de ejecutar varias tareas a la vez, compartiendo la computadora misma y los recursos de cómputo entre los programas, obteniendo una mayor optimización en las actividades que desempeña.

La técnica de **Multitareas** ha estado durante mucho tiempo en las grandes y medianas computadoras y ahora en las microcomputadoras tiene mucha demanda y aceptación.

Se han desarrollado muchas y diferentes estrategias de **Multitareas**, cada una de las cuales optimiza los recursos y aspectos que son más importantes para el diseñador de sistemas.

Si existen suficientes tareas en memoria, es posible mantener a la unidad de procesamiento central (UPC) la mayor parte del tiempo ocupada, optimizando el uso de este recurso.

Esta forma de **multitareas** llamada multiprogramación proporciona muchas ventajas, pero esta habilidad tiene un costo: Aunque más programas puedan estar corriendo al mismo tiempo, cada uno tomará más tiempo para terminar.

Un sistema de multitareas debe, por su naturaleza propia, ser diseñado interactuando intrínsecamente con el sistema operativo a través de las llamadas normales al sistema o modificándolas para permitir la conmutación entre tareas.

Desde el punto de vista del usuario, el **sistema de multitareas** parece virtualmente ser el sistema operativo. Esto es debido a que el **sistema de multitareas** provee todos los servicios de interfaz que maneja el sistema operativo tradicional. Desde el punto de vista del programador, el **sistema de multitareas**, es una potente caja de servicios interconectados que permiten la ejecución de varios programas a la vez, sin que estos sufran modificación alguna para poder correr.

Uno de los objetivos del **sistema de multitareas** es hacer que el usuario, que tiene familiaridad básica con el sistema operativo "MS-DOS", pueda correr varias aplicaciones sin ningún entrenamiento a priori.

El concepto de Multitareas ha llegado a ser muy popular en sistemas operativos que anteriormente eran realizados en mini o superminicomputadoras, como UNIX y que ahora se han adaptado para microcomputadoras también, pero al costo de obtener una respuesta lenta, necesitando de procesadores poderosos como el 80386 de INTEL para obtener una respuesta aceptable.

Compilar un programa, descargar un archivo a disco y ordenar una larga tabla de una base de datos tienen algo en comun; acapararán el uso de la computadora por un tiempo.

Para la mayoría de nosotros que trabajamos con computadoras personales IBM y/o compatibles, es desagradable tener que esperar a que finalice el proceso que acaparó el uso de la computadora, volviendo algunos procesos inoperantes al permanecer mucho tiempo inactivos. **Multitareas** nos permite continuar trabajando sobre la computadora mientras otras tareas están siendo realizadas por ella.

Las computadoras personales que utilizan el ampliamente aceptado sistema operativo MS-DOS **no** proporcionan ésta capacidad, hasta hoy en día.

En general, observamos que el uso de las computadoras personales abarcan un gran número de actividades y empieza en algunas otras con un gran potencial de aplicaciones.

Las encontramos en tiendas de autoservicio, bancos, comercios, industria, etc. Sin embargo, a pesar de que sabemos estas aplicaciones, tenemos también que aceptar que el potencial de programadores e ingenieros en sistemas nacionales, no se han incorporado al desarrollo en este tópico. La situación anterior motivó a realizar investigación para desarrollar un sistema que permita conjuntar varios aspectos básicos.

El objetivo del sistema es proporcionar una herramienta básica de operación consistente de un **sistema de Multitareas en tiempo real con un ambiente de ventanas para la computadora IBM o 100% compatibles para ejecutarse sobre la cima del sistema operativo MS-DOS.**

El trabajo se desarrolló en un sistema IBM PC-XT con 640K bytes de memoria, donde 1K = 1024 bytes, utilizando las herramientas de desarrollo Turbo-C, Turbo Ensamblador y Turbo Depurador de Borland.

En los capítulos siguientes se detallará la estructura del sistema:

En el II.- Se presentan las principales características de programación de la IBM PC-XT, el mapa de la memoria convencional de la computadora personal IBM, el sistema operativo que comprende al sistema básico de entradas y salidas (BIOS) y al sistema operativo MS-DOS, acceso a los sevicios del sistema, manejo de interrupciones, llamadas al BIOS y llamadas al MS-DOS.

En el III.- Se presenta la teoría básica de sistemas operativos.

En el IV.- Se presenta el tipo de comunicación serie asíncrona.

En el V.- Se describe el diseño del sistema de multitareas.

En el VI.- Conclusiones y recomendaciones.

II.- CARACTERISTICAS DE PROGRAMACION PRINCIPALES DE LA IBM-PC.

Un sistema de cómputo puede dividirse en forma general en cuatro componentes.

- La circuitería (UPC, memoria, dispositivos de Entrada-Salida).
- El sistema operativo.
- Los programas de aplicación (Compiladores, sistemas de bases de datos, juegos de video, programas de negocios).
- Los usuarios (Personas, máquinas u otras computadoras).

La circuitería proporciona los recursos básicos de cómputo. Los programas de aplicación definen la manera en la cual estos recursos serán utilizados para resolver los problemas de los usuarios. El sistema operativo controla y coordina el uso de la circuitería por los programas de aplicación.

II.1.- Mapa de memoria de la IBM-PC (XT).

La memoria es una de las áreas más importantes de la computadora y debe ser bien comprendida. La figura 1 muestra la distribución de la memoria convencional, término empleado para un direccionamiento directo de hasta un megabyte (1,048,576 bytes) de memoria por un microprocesador Intel 8086/8088 o por un microprocesador 80286 o 80386 corriendo en modo real (los cuales actúan como un 8086-8088), (Jourdain, R., 1988).

Las direcciones relativas referenciadas por el microprocesador dentro de la memoria convencional, son generadas por un registro de segmento de 16 bits, el cual actua como un registro base y contiene una dirección de todo un párrafo o segmento de 64K bytes de largo, podiendo direccionar hasta dieciseis segmentos, combinado con un registro índice de 16 bits, el cual actúa como un desplazamiento que apunta a la dirección exacta del byte dentro del segmento.

Sobre las computadoras personales IBM y compatibles, la tabla de los vectores de interrupción, área de datos del BIOS, el sistema operativo tradicional MS-DOS y los programas de aplicación que corren bajo su control ocupan los 640 Kbytes (KB) de la parte baja del espacio de memoria convencional.

El espacio por encima de los 640 KB esta dividido para los circuitos que utilizan la memoria de solo lectura (ROM), los cuales se encuentran en la tarjeta principal de circuitería, espacio para el sistema de video, manejadores del disco fijo y para programas de prueba. Además espacio para circuitos basados en memoria de lectura y escritura (RAM) y circuitos basados en ROM que se encuentran sobre las tarjetas de expansión que son usadas para operaciones de entrada-salida y por rutinas de dispositivos adicionales independientes.

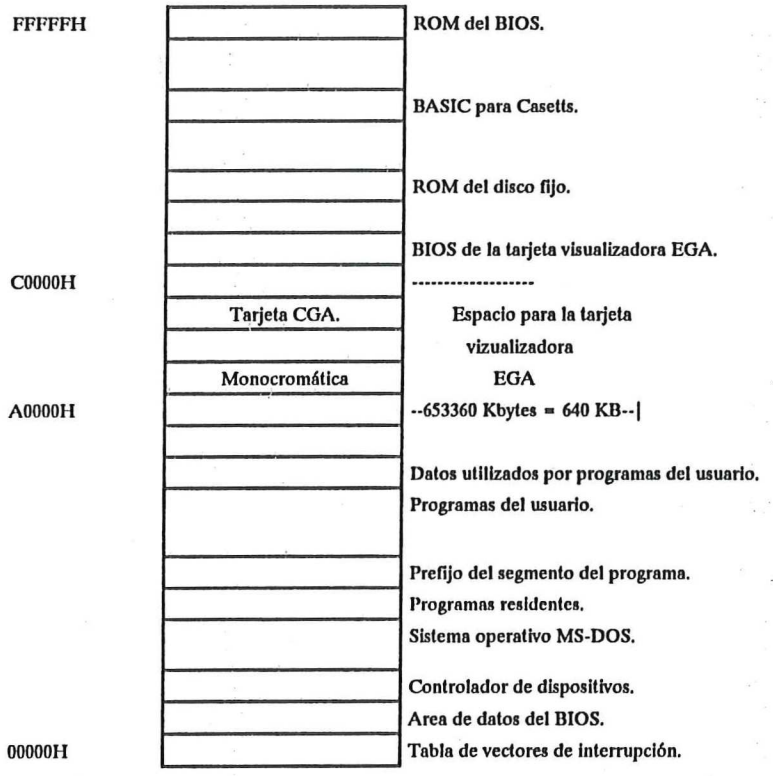

Figura 1.- Distribución de la memoria convencional (1 Mega byte de espacio direccionable) de la IBM.

Los 640 KB de la parte baja de la memoria son administrados por el sistema operativo MS-DOS y se divide en tres zonas:

1) La tabla de vectores de interrupción.

2) El área del sistema operativo.

3) El área de programas transitorios.

La tabla de vectores de interrupción ocupa los 1024 bytes más bajos de la memoria (localidades 00000H-003ffH). La letra "H" indica base hexadecimal y se emplearán indistintamente direcciones absolutas y relativas. Cada posición de palabras dobles (4 bytes) en la tabla es llamada vector y contiene la dirección (segmento y desplazamiento) de una rutina de atención asociada a un número de interrupción.

El área del sistema operativo inicia inmediatamente sobre la tabla de los vectores de interrupción y contiene el área de datos del BIOS, el sistema operativo tradicional, sus tablas y espacios de memorias y manejadores de dispositivos adicionales, especificados en los archivos del sistema (los archivos .SYS) y la porción residente del intérprete de mandos COMMAND.COM.

El área de programas transitorios y del usuario es el resto de la memoria RAM sobre el área del sistema operativo, extendiéndose hasta el límite de los 640 KB de RAM instalada.

El área de memoria por encima de los 640 KB es subdividida para funciones particulares del sistema.

Los 128 KB siguientes están reservados como RAM de video, usados por los adaptadores de video. Este espacio es donde texto y gráficos residen cuando son proyectados sobre pantalla.

Los 128 KB siguientes están reservados para los programas de control, que estan en (ROM). Los programas son almacenados sobre pastillas especiales que han sido fusionados a los circuitos de tal forma que no pueden alterarse. Tarjetas de graficado, controladores de disco duro, tarjetas de comunicación y tarjetas de memoria expandida son ejemplos que pueden usar parte de la memoria ROM.

Finalmente los últimos 128 Kbytes de memoria estan reservados para el ROM sobre la tarjeta principal. Este espacio es donde el BIOS reside, así como el programa de auto prueba de inicialización (POST) y el cargador (bootstrap). Estos programas son la prueba maestra y programas de control para el sistema. Habilitan al sistema para que cargue al sistema operativo que se encuentra, ya sea en disco flexible o disco duro.

II.2.- Sistema operativo.

Todos sabemos que una computadora por si misma, por más avanzada y completa que esta sea, es de poca o nula utilidad sin los programas que le den la capacidad que le permita desarrollar las actividades para las cuales se desea utilizar.

La programación de la computadora puede dividirse de modo general en dos clases; los programas del sistema que manejan la operación de la computadora misma y los programas de aplicación que resuelven problemas del usuario. El fundamental de todos los programas es el sistema operativo, que controla todos los recursos de la computadora y ofrece la base sobre la cual pueden desarrollarse y ejecutarse los programas de aplicación.

Un sistema operativo es un conjunto de programas supervisores interrelacionados que manejan y controlan el procesamiento de la computadora. Los programas del sistema operativo actuan como una interfaz entre los usuarios de la computadora y la circuitería de la computadora. El propósito principal de un sistema operativo es proporcionar un ambiente en el cual los usuarios puedan ejecutar sus programas de aplicación sin tener que involucrarse con el manejo de la circuitería de la computadora. En general un sistema operativo provee lo siguiente:

a) Manejo de almacenamiento.

b) Manejo de procesamiento.

c) Seguridad.

e) Interfaz con el humano.

La clasificación de los sistemas operativos existentes para microcomputadoras segun la enciclopedia del MS-DOS se ubican en tres principales categorias: Monitores ROM, sistemas operativos tradicionales y ambientes de operación.

Las características generales de éstas tres categorias se enlistan en la tabla I. (The MS-DOS Encyclopedia, 1988).

Tabla I.- Características de los tres principales tipos de sistemas operativos.

	Monitores ROM	Sistemas operativos tradicionales.	Ambientes operacionales
Complejidad:	Baja.	Mediana.	Alta.
Construido sobre:	Circuitería	BIOS.	Sist. oper.
Distribuidos en:	ROM.	Disco.	Disco.
Programados sobre:	ROM.	Disco.	Disco.
Apoyo de periféricos:	Física.	Lógica.	Lógica.
Acceso a disco mediante:	Sector.	Sistema de archivos	Sist.de arch.
Ejemplo:	BIOS ROM	MS-DOS.	Windows

Un monitor ROM, que puede ser el sistema operativo básico de entrada-salida (BIOS) es el tipo más simple de sistema operativo. Este es diseñado para alguna configuración particular de circuitería y provee programas básicos de acceso a periféricos conectados a la computadora.

El sistema operativo tradicional para microcomputadoras es construido en la cima de uno del tipo monitor ROM. Provee características adicionales tales como un sistema de archivos y acceso lógico a periféricos. El acceso lógico a periféricos permite que los programas corran independientemente de la circuitería. Un sistema operativo tradicional también almacena programas en archivos o dispositivos de almacenamiento externos (discos, disketts, etc..) y a una solicitud del usuario o programa, los carga en memoria principal para su ejecución. El sistema operativo MS-DOS es un sistema operativo tradicional.

Un sistema operativo de ambiente (o ambiente de operación) es construido en la cima del sistema operativo tradicional. Provee servicios adicionales, tales como menus, formas de apoyo en ventanas y otras utilerias, que simplifica y optimiza la operación de los programas sobre la computadora. El sistema que se desarrolló en éste trabajo cae en esta categoría.

II.2.1.- Sistema básico de entradas y salidas (BIOS).

Para poder comunicarse con la computadora desde el momento en que se enciende, empiezan a realizarse una serie de pasos perfectamente establecidos. Muchas de las acciones

que realiza la computadora están directamente ligadas, exclusivamente con la circuitería. Muchas también, están de antemano bien definidas en códigos de instrucciones abreviadas de una manera perfectamente determinada.

Se encuentran por un lado, las del sistema básico de entrada-salida (BIOS) que vienen grabadas en los circuitos integrados ROM. Esta pastilla se encuentra en la tarjeta principal de la computadora y es proporcionada por el fabricante. El BIOS se encuentra dentro de la clasificación de monitores ROM. El objetivo de estas subrutinas es el proveer al usuario una interfaz más comprensible entre los detalles de la circuitería y el programador, evitándole la penosa tarea de tratar con ella directamente.

Por otro lado existe un código de instrucciones a un nivel más alto (se entiende nivel más alto al hecho de que las instrucciones se entienden más fácilmente por el humano que está dedicado a la programación o uso de la computadora), que el BIOS y a éste en conjunto con el BIOS se le llama sistema operativo.

El BIOS es quien provee a nivel dispositivo el control de las principales operaciones de entrada-salida de los circuitos que se encuentran en el sistema. Opcionalmente hay módulos adicionales de ROM a nivel BIOS que pueden ser encontrados en las tarjetas adaptadoras que proveen control para ellas.

Las rutinas del BIOS proporcionan al programador a nivel ensamblador la posibilidad de realizar operaciones de entrada-salida de bloques de información (a disco flexible y a disco duro) o por caracteres, sin se vea involucrado con las direcciones y características de los dispositivos. Provee servicios al sistema, servicios de video, de teclado, a discos, a impresora, de comunicación, etc. Además permite que nuevos dispositivos puedan ser agregados al sistema manteniendo la interfaz a nivel del BIOS para estos dispositivos. De esta manera los programas llegan a ser transparentes a las modificaciones de la circuitería.

El acceso al BIOS es a travéz de las interrupciones por programación. Cada punto de entrada al BIOS esta disponible mediante un vector de interrupción, (dirección que indica el lugar donde se encuentra la rutina de atención para la interrupción presentada).

Todos los parámetros desde y hacia el BIOS van a través de los registros del microprocesador. Si una función del BIOS tiene varias operaciones posibles, el registro AH es utilizado como entrada para indicar la operación deseada. Generalmente las rutinas del BIOS salvaguardan todos los registros excepto el registro AX y el registro de banderas, salvo que se indique explícitamente la modificación de algún otro registro como parte del resultado al llamarlas (IBM technical reference, 1986).

Todos los microprocesadores 8086/8088 y compatibles inicializan su ejecución con los registros de código de segmento (CS) y registro apuntador de programa (IP), apuntando a la dirección FFFF:0000H, la cual típicamente tiene una instrucción de transferencia incondicional hacia una dirección cuyo destino es el ROM BIOS que contiene el código de inicialización para la máquina. La figura 2 muestra la distribución de la memoria convencional al inicializar el sistema.

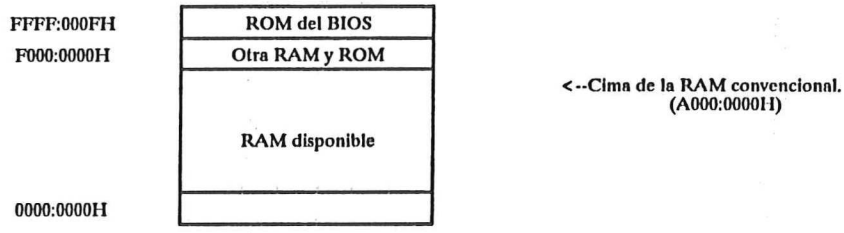

Figura 2.- Distribución de la memoria convencional al inicializar el sistema.

El ROM BIOS ahora inicia el procedimiento de cargado-inicializador (bootstrap) mediante la ejecución de la rutina cargador del ROM.

Todas las interrupciones del microprocesador están deshabilitadas cuando inicia su operación y permanecen así hasta que se ejecuta la rutina de formación de la tabla de vectores de interrupción en la base de la memoria.

La rutina de inicialización en el ROM BIOS (el procedimiento POST) usualmente determina y prueba qué dispositivos estén instalados así como la memoria instalada.

Cuando se determina que la computadora está en posibilidad de operación, el ROM BIOS realiza normalmente las siguientes acciones. Primero inicializa la tabla de vectores de interrupción al inicio de la memoria y a cualquier controlador de interrupciones que haga

referencia a esta tabla. En seguida, el ROM BIOS prepara cualquier interfaz necesaria, tales como los controladores de acceso directo a memoria (CADM), puertos serie, etc. Revisa si existen ROM BIOS adicionales instalados. Si se encuentran, sus rutinas de inicialización son llamadas para que inicialicen los dispositivos asociados.

Por ejemplo existen ROM adicionales para el ROM BIOS del disco fijo y para el adaptador de tarjetas de gráficas extendidas (EGA). La figura 2.1 muestra la distribución de la memoria convencional con la tabla de vectores de interrupción y las tablas del ROM BIOS.

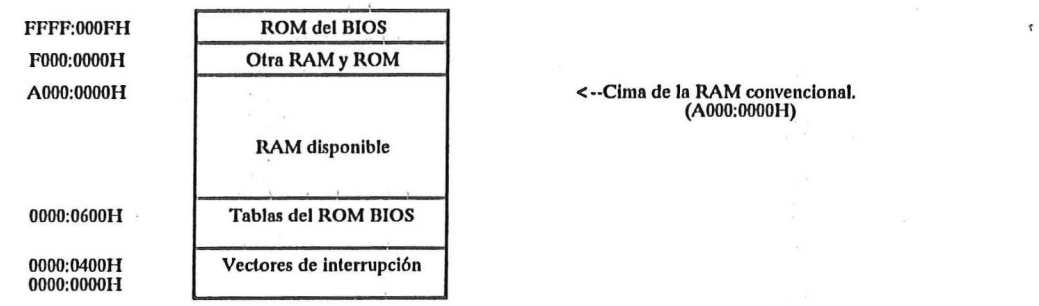

Figura 2.1.- Distribución de la memoria convencional con las tablas de interrupción y las tablas del ROM BIOS.

Esta rutina revisa primero el manejador de disco flexible para ver si hay un disco con el cargador del sistema operativo tradicional. Si no lo hay, entonces invoca al ROM asociado con otro dispositivo que lo pueda tener (disco fijo). Este procedimiento se repite hasta que se encuentra el cargador.

En caso de que no se encuentre, entonces el programa BASIC en ROM es habilitado. Cuando es encontrado el cargador del sistema operativo se le pasa el control a éste. A su vez el cargador del sistema operativo utiliza los servicios del ROM BIOS a través de la tabla de interrupciones para cargar la siguiente parte del sistema operativo. La figura 2.2 muestra la distribución de la memoria convencional cuando es cargado el sector del cargador del sistema operativo.

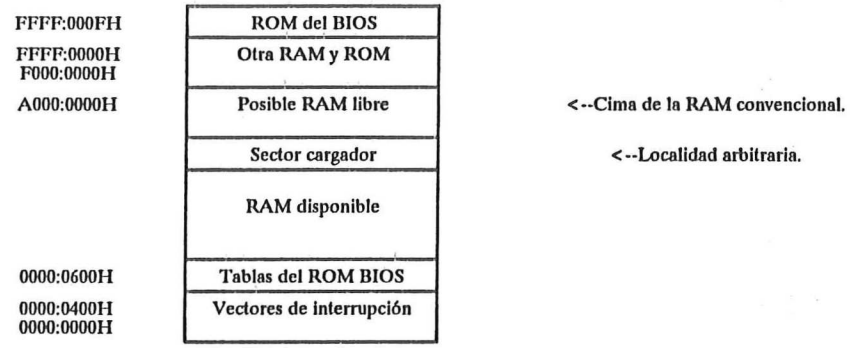

Figura 2.2.- Distribución de la memoria convencional cuando es cargado el sector del cargador del sistema operativo.

Antes de continuar, el cargador del sistema operativo lee la configuración del disco que se encuentra en una estructura de datos llamada el bloque de parámetros del BIOS (BPB). La figura 2.3 describe una posible configuración del BPB. A partir de esta información, el cargador del sistema operativo calcula la localidad de la raíz del directorio y puede verificar que las dos primeras entradas de la raíz son los archivos del sistema IO.SYS y MS-DOS.SYS.

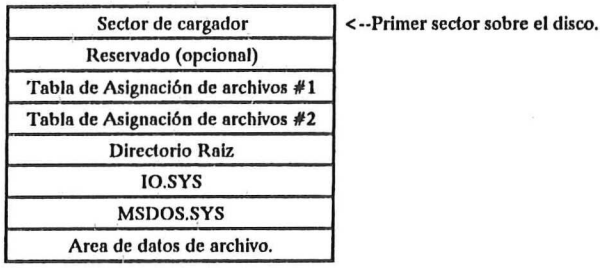

Figura 2.3 Una posible configuración del bloque de parámetros del BIOS (BPB).

II.2.2.- Sistema operativo MS-DOS.

El sistema operativo tradicional MS-DOS, provee un conjunto de programas interrelacionados que controlan y manejan el procesamiento de la computadora personal IBM. Los módulos principales del MS-DOS son: El BIOS del MS-DOS (IO.SYS), el núcleo (MSDOS.SYS) y el procesador de mandos (COMMAND.COM).

El archivo IO.SYS está dividido internamente por tres módulos: Los manejadores de dispositivos residentes, el módulo básico de inicialización BIOS del MS-DOS y el módulo de inicialización SYSINIT del sistema MS-DOS.

El conjunto mínimo de manejadores residentes son: Para el teclado y visualizador (CON), para la impresora (PRN), usualmente un puerto paralelo, para el dispositivo de entrada-salida auxiliar (AUX), usualmente un puerto serie, para el reloj en tiempo real (CLOCK$) y el controlador para un dispositivo de un bloque (varia de A al E).

Para llegar a tener el ambiente que proporciona el MS-DOS, una serie de procesos deben de realizarse.

La inicialización del sistema MS-DOS comienza después de que el cargador del sistema operativo del BIOS ha cargado a los archivos IO.SYS y MSDOS.SYS y se ha transferido el control al archivo IO.SYS. La figura 2.4 muestra la distribución de la memoria convencional cuando han sido cargados archivos IO.SYS y MSDOS.SYS.

Hasta este momento no han sido impuestos procedimientos estandares por parte del MS-DOS. Sin embargo cuando el control ha sido transferido al IO.SYS, el MS-DOS impone sus estándares.

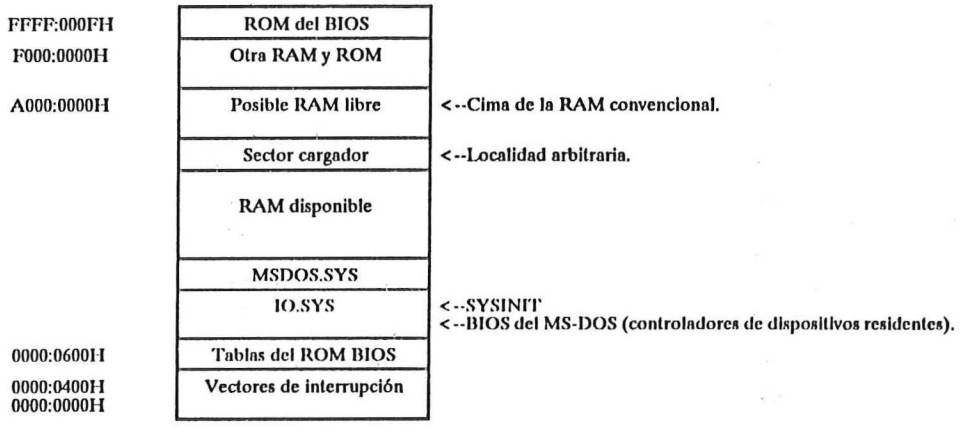

Figura 2.4.- Distribución de la memoria convencional cuando han sido cargados los archivos del sistema IO.SYS y MSDOS.SYS

Los dos módulos de inicialización son usualmente descartados tan pronto como el MS-DOS este completamente inicializado y los programas del procesador de mandos esten

corriendo; los manejadores de dispositivos residentes permanecen en memoria principal mientras que el MS-DOS este corriendo y por lo tanto son colocados en la primera parte del archivo IO.SYS antes de la inicialización de los módulos.

El módulo de inicialización BIOS del MS-DOS comunmente despliega un mensaje de los derechos reservados por el fabricante y notifica que ha creado el archivo IO.SYS. En todas las computadoras IBM y compatibles, examina las entradas que se encuentran en la tabla de vectores de interrupción para determinar qué dispositivos fueron encontrados por el ROM BIOS (cuando se ejecutó la rutina de prueba de auto encendido -POST-) y ajusta la lista de manejadores de dispositivos residentes. La rutina de inicialización del BIOS del MS-DOS esencialmente termina aquí y el control es transferido al módulo SYSINIT.

El módulo SYSINIT localiza la parte superior de la memoria RAM disponible y se copia ahí a si mismo. Entonces este transfiere el control a la copia y ésta continua con la inicialización del sistema. El primer paso es mover al archivo MSDOS.SYS, a una posición inmediatamente en seguida de la porción final del IO.SYS. Este movimiento sobre escribe encima de la copia original del SYSINIT y usualmente sobre toda la rutina de inicialización del BIOS del MS-DOS, los cuales ya no son necesarios. La figura 2.5 muestra la distribución de la memoria convencional cuando el cargador SYSINIT y el archivo MSDOS.SYS han sido reubicados.

El módulo SYSINIT llama a la rutina de inicialización del núcleo del MS-DOS. Esta rutina realiza la construcción interna del núcleo, incluyendo la colocación adecuada de los valores de los vectores para las interrupciones de la 20H a la 3Fh.

La rutina de inicialización del núcleo del MS-DOS llama a las funciones de cada uno de los manejadores residentes para formar los vectores para cualesquier interrupción externa por circuitería utilizada por los dispositivos. Cada dispositivo manejador de bloque regresa un apuntador hacia un bloque de párametros del BIOS (BPB) para cada manejador que soporta; estos BPB son inspeccionados por el módulo SYSINIT para encontrar el tamaño de sector más grande utilizado por alguno de los manejadores.

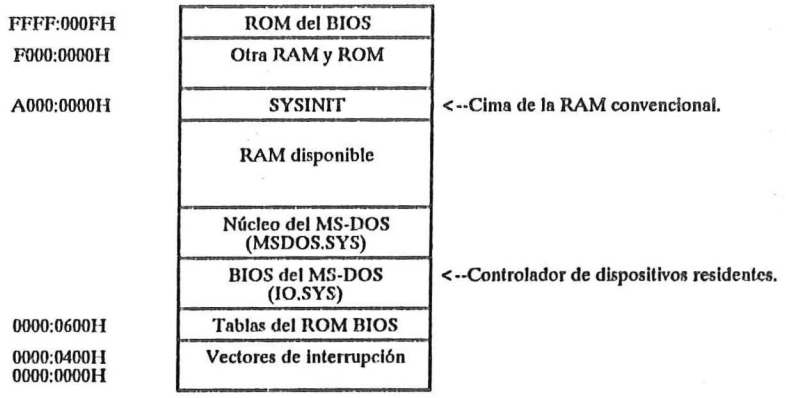

Figura 2.5.- Distribución de la memoria convencional cuando el cargador SYSINIT y el archivo MSDOS.SYS han sido reubicados.

La rutina de inicialización del núcleo entonces asigna un espacio de memoria al sector más grande encontrado y coloca el manejador de dispositivo NUL en la cabeza de la lista de manejadores de dispositivos.

La rutina de inicialización del núcleo antes de regresar el control al SYSINIT despliega un mensaje de derechos reservados del MS-DOS. El cargado de esta porción del sistema del MS-DOS ha sido completada y el módulo SYSINIT puede utilizar cualquiera de las funciones del MS-DOS en conjunto con el grupo de manejadores de dispositivos residentes.

Lo siguiente que hace el módulo SYSINIT es intentar abrir el archivo CONFIG.SYS que comúnmente se encuentra en la raiz del directorio del disco que fue cargado. Si el archivo no existe, SYSINIT utiliza los parámetros estandares del sistema; si el archivo se encuentra, entonces el modulo SYSINIT lee completamente las indicaciones del archivo y encuentra la información tal como el número de espacio de memoria asignados a los discos, el número de entradas en la tabla de archivos y el número de entradas en la tabla del manejador de traducciones. A todas estas estructuras se les asigna un espacio de memoria (tablas del MS-DOS) justamente encima del núcleo del MS-DOS.

El SYSINIT procesa el texto del archivo CONFIG.SYS secuencialmente para determinar que manejadores de dispositivos serán implantados. Carga los bloques de control de archivos

de los manejadores de dispositivos en memoria, después de los espacios de memoria de disco del sistema y de los archivos y tablas del MS-DOS. Y en seguida carga a los manejadores de dispositivos instalables. La figura 2.6. muestra la distribución de la memoria convencional cuando han sido cargados las tablas de asignación de archivos del MS-DOS, las direcciones de memoria reservada a disco, los bloques de control de archivo (BCA) y los manejadores de dispositivos instalables.

El módulo SYSINIT procede a cerrar todos los archivos abiertos y entonces abre los manejadores de dispositivo CON, PRN y AUX.

La última función del módulo SYSINIT es cargar y ejecutar al procesador de mandos (COMMAND.COM).

El procesador de mandos es cargado al inicio de la memoria libre justamente después de los manejadores de dispositivos instalables o después del último bloque de control de archivos si es que no hay manejadores de dispositivos instalables. La figura 2.7 muestra la distribución de la memoria convencional cuando ha sido cargado el procesador de mandos.

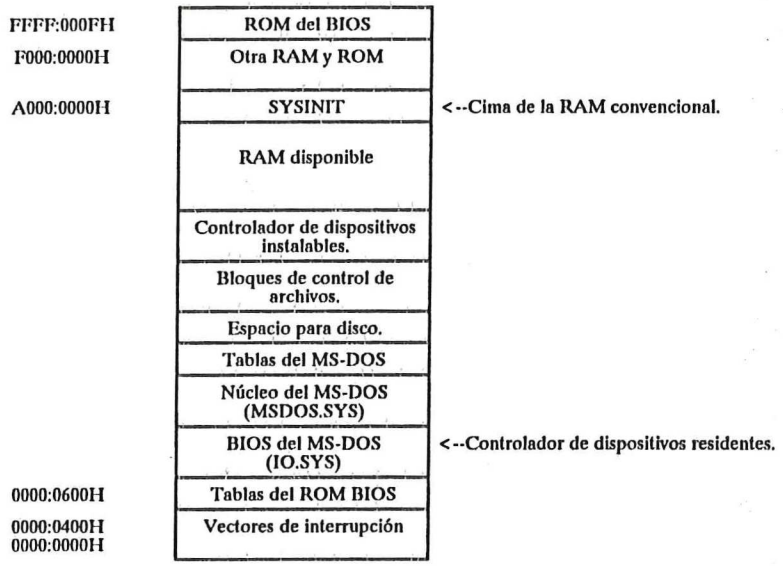

Figura 2.6.- Distribución de la memoria convencional con tablas de asignación de archivos del DOS, memoria reservada a disco, BCA y controladores de dispositivos instalables.

El procesador de mandos COMMAND.COM consiste de tres partes: Una porción residente, un módulo de inicialización y una porción transitoria.

La porción residente contiene el apoyo para la terminación de los programas iniciados por el COMMAND.COM que se encuentra en memoria sobre los manejadores de dispositivos instalables.

El módulo de inicialización se encuentra en memoria encima del módulo residente. Este es llamado solo una vez por la porción residente. Primero mueve a memoria una copia provisional del COMMAND.COM (porción transitoria), encima de el mismo. Entonces procesa la información especificada en el comando "shell" que se encuentra en el archivo CONFIG.SYS, si es que hay alguno.

En seguida procesa el archivo AUTOEXEC.BAT si existe, y finalmente transfiere el control a la porción residente, el cual libera el espacio de memoria utilizado por el módulo de inicialización y la porción transitoria del COMMAND.COM.

La porción residente despliega una señal para el usuario indicando que está listo para aceptar mandos.

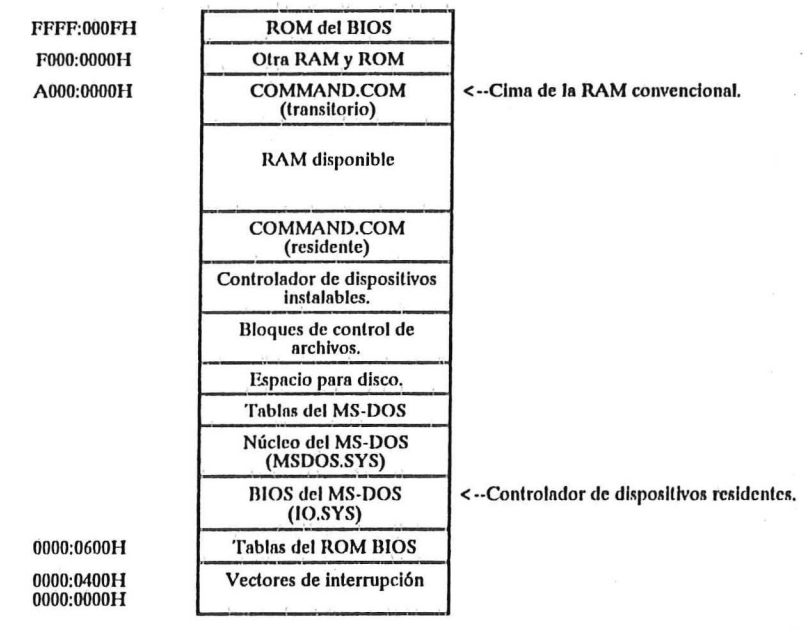

Figura 2.7.- Distribución de la memoria convencional cuando ha sido cargado el procesador de mandos COMMAND.COM.

II.3.- Accesos a los servicios del sistema.

La interfaz entre el sistema operativo y los programas del usuario está definida por un conjunto de instrucciones extendidas que proporciona el sistema operativo. A estas instrucciones extendidas se les conoce como llamadas al sistema. Muchas de las funciones primitivas que proporciona el sistema tienen su acceso a través de interrupciones.

Las interrupciones son rutinas ya preparadas que contiene el sistema y que son llamadas para realizar alguna tarea común.

II.3.1.- Manejo de interrupciones.

Hay dos tipos de interrupciones; interrupciones por circuitería e interrupciones por programación.

Las interrupciones por circuitería son las causadas por señales generadas por piezas electrónicas que se encuentran dentro de la computadora, ya sea sobre la tarjeta del sistema o desde una tarjeta que se encuentra en un canal de entrada-salida. Por ejemplo pueden producirse por un pulso del circuíto cronómetro contador (timer), por una señal proveniente de la impresora, la presencia de una acción sobre el teclado, o por muchos otros motivos de índole de circuitería.

Existen ocho niveles de prioridad para las ocho interrupciones por circuitería del sistema de la computadora IBM-PC XT (dieciseis para la IBM-AT), seis de las cuales son canalizadas a las ranuras de expansión de la tarjeta principal del sistema para usos de las tarjetas adaptadoras. Dos niveles son utilizados sobre la tarjeta principal del sistema. El nivel "0", el de más alta prioridad, es asignado al canal 0 del cronómetro contador y el nivel "1" es asignado a los circuitos adaptadores del teclado.

Las direcciones de las interrupciones se llaman vectores de interrupción. Cada vector tiene una longitud de 4 bytes. Los dos primeros bytes contienen la parte baja (segmento de código) y los dos segundos bytes la parte alta (apuntador de instrucción) de la dirección de la rutina que van a realizar.

Los últimos 1024 bytes de memoria (parte baja de la memoria) contienen a los vectores de interrupción, así que hay espacio para un total de 256 vectores.

Cuando una interrupción por circuitería es llamada, la UPC detiene lo que esta haciendo, atiende la interrupción y entonces regresa a continuar con lo que dejó suspendido. Para ser capaz de regresar al punto exacto siguiente, en el que fué interrumpida, la UPC salvaguarda el registro de banderas, el registro de segmento de código (CS) y el registro apuntador de instrucciones (IP) en pila. Entonces carga el vector de interrupción dentro de los registros CS:IP dándole luego el control a la rutina de interrupción.

Toda rutina de interrupción termina con una instrucción IRET, la cual reemplaza los valores de los registros CS:IP por los valores iniciales de los mismos antes de que se efectuara una interrupción, asi como el registro de banderas, obteniéndolos de la pila donde fueron salvaguardados. Así la UPC puede continuar con el programa que se encontraba ejecutando.

Las interrupciones por programación son causadas por la ejecución de la instrucción "int" del microprocesador. Estas son procedimientos que los programas pueden llamar para realizar alguna tarea común. Sin embargo, éstas rutinas se encuentran en el sistema operativo y no dentro de los programas del usuario. El término interrupción es por el método mediante el cual se pueden anexar. Las interrupciones por programación, a diferencia de las interrupciones por circuitería no pueden tomar el control durante una interrupción por circuitería.

II.3.2.- Llamadas al BIOS.

El BIOS divide a las interrupciones en cinco grupos:

1.- Los vectores de interrupción para el microprocesador 8088/86 (de la 0H a la 07H), (Tabla II).

2.- Los vectores de interrupción para el controlador de interrupciones programable 8259-A (de la 08H a la 0FH), (Tabla III)

3.- Puntos de entrada al BIOS (de la 10H a la 1AH y 40H), (Tabla IV).

4.- Interrupciones proporcionadas al usuario (1BH y 1CH), (Tabla V).

5.- Apuntadores a tablas de datos del BIOS (1DH, 1EH, 1FH y 41H), (Tabla VI), (Scanlon, L. J.;1985).

Tabla II.- Vectores de interrupción para el microprocesador 8088/8086

Interrupción.	Dirección.	Nombre.
00H	00H-3H	División por cero.
01H	04H-7H	Ejecución de instrucciones paso a paso.
02H	08H-BH	No enmascaramiento.
03H	0CH-FH	Punto de ruptura.
04H	10H-13H	Sobreflujo.
05H	14H-17H	Vaciado de pantalla a impresora.
06H	18H-1BH	Reservado.
07H	1CH-1FH	Reservado.

Tabla III.- Vectores de interrupción para el controlador de interrupciones 8259-A.

Interrupción.	Dirección.	Nombre.
08H	20H-23H	Cronómetro.
09H	24H-27H	Teclado.
0AH	28H-2BH	Reservado.
0BH	2CH-2FH	Comunicación.
0CH	30H-33H	Comunicación.
0DH	34H-37H	Disco fijo.
0EH	38H-3BH	Diskette.
0FH	3CH-3FH	Impresora.

Tabla IV.- Puntos de entrada al BIOS.

Interrupción.	Dirección.	Nombre.
10H	40H-43H	Servicios de video.
11H	44H-47H	Revisión de equipo instalado.
12H	48H-4BH	Tamaño de memoria presente.
13H	4CH-4FH	Servicios del disco.
14H	50H-53H	Servicios de comunicación.
15H	54H-57H	Servicios de casettes.
16H	58H-5BH	Servicios de teclado.
17H	5CH-5FH	Servicios de impresora.
18H	60H-63H	BASIC residente.
19H	64H-67H	Cargador "bootstrap".
1AH	68H-6BH	Hora y día.

Tabla V.- Interrupciones proporcionadas al usuario.

Interrupción.	Dirección.	Nombre.
1DH	74H-77H	Inicialización de video.
1BH	6CH-6FH	Interrupción del teclado (Ctrl-Break).
1CH	70H-73H	Señal del cronómetro.

Tabla VI.- Apuntadores a tablas de datos del BIOS.

Interrupción.	Dirección.	Nombre.
1DH	74H-77H	Inicialización de video.
1EH	78H-7BH	Parámetros del diskette.
1FH	7CH-7FH	Carta de gráficos de video.
41H	104H-107H	Parámetros de disco fijo.
5AH	168H-16BH	Agrupamientos (clusters).
5BH	16CH-16FH	Usado por agrupamientos (cluster) de programas.
60H-67H	Variable.	Reservado para programas de usuario.

Generalmente cada rutina de interrupción está diseñada cuidadosamente para salvaguardar todos los registros, reestableciéndolos al concluir. De tal manera que no deja rastro alguno de que haya ocupado la UPC. Solo aquellos registros en los que se indique explícitamente algun cambio como respuesta a la petición de la rutina que lo llamó.

II.3.3.- Llamadas al sistema operativo MS-DOS.

IBM reserva los tipos de interrupción de la 20H a la 3FH, para ser utilizados por el MS-DOS. De los cuales solamente de la 20H a la 27H están actualmente implantadas.

La mayoría de los servicios que provee el MS-DOS generalmente se invocan a través de la interrupción 21H. Otros son llamados utilizando las interrupciones 20H y de la 22H a la 27H y 2FH.

La interrupción 21H se emplea en vez de las rutinas específicas del BIOS y ayudan a asegurar que el programa corra sobre cualquier computadora y BIOS que este corriendo adecuadamente al MS-DOS, ya que se supone que cada versión nueva del MS-DOS es compatible con las anteriores.

Las llamadas a las funciones de la interrupción 21H son comunmente invocadas como sigue:

Se establece en el registro AH = Número de función.

En el registro AL = Código de la subfunción (si se requiere).

Los otros registros = Información adicional para la función específica.

Se ejecuta la interrupción 21H.

Las llamadas a las funciones de la interrupción 21H se caracterizan de acuerdo al servicio que proporcionan, las cuales comunmente se agrupan por el tipo de tareas que realizan, estas son:

a) Funciones de entrada de caracteres.

b) Funciones de salida de caracteres.

c) Funciones de manejo de disco.

d) Funciones de manejo de archivos.

e) Funciones de manejo de sartas de caracteres con acceso secuencial, aleatorio, absoluto y relativo.

f) Funciones de manejo de directorio.

g) Manejo de procesos.

h) Manejo de memoria.

i) Manejo misceláneo del sistema.

Los programas de aplicación tienen acceso a las funciones del sistema utilizando instrucciones de interrupción por programación.

Por otro lado se tienen las demás interrupciones que proporcionan las siguientes operaciones (Tabla VII):

Tabla VII.- Interrupciones proporcionadas por el MS-DOS.

Interrupción.	Nombre.
20H	Finaliza programas.
21H	Funciones descritas anteriormente.
22H	Dirección de finalización de rutina.
23H	Dirección del manejador del Ctrl-C.
24H	Dirección del manejador de errores críticos.
25H	Lectura absoluta a disco.
26H	Escritura absoluta a disco.
27H	Termina y mantiene residente a programas.
28H-2EH	Reservadas.
2FH	Multicanalización
30H-3FH	Reservadas.

III.- TEORIA DE SISTEMAS OPERATIVOS.

Históricamente los sistemas operativos se han desarrollado para satisfacer dos necesidades, que algunas veces son complementarias y conflictivas.

La primera es para manejar los recursos del material (circuitería) e información (programas) de un sistema de cómputo que asegure su eficiencia y correcta operación.

La segunda es proporcionar al usuario los servicios que esencialmente transforman la colección de los componentes de una máquina a un lenguaje de alto nivel en una computadora virtual más adecuada para su uso. (Shaw, A., 1988).

III.1.- Procesos.

El concepto central en sistemas operativos es el de proceso.

Un proceso es un programa en ejecución.

Un proceso generalmente tiene tres partes: Una sección administrativa, una sección para las instrucciones del programa y una sección para los datos y variables globales (las variables locales de una subrutina son normalmente puestas sobre la pila, en el área administrativa del proceso, a menos que sean variables estáticas).

Siempre que un programa se ejecuta, el sistema operativo crea un proceso en memoria. El proceso contiene el programa (código de instrucciones de máquina y el espacio para datos), que vienen siendo respectivamente la sección de las instrucciones del programa y sección de datos y la información requerida por el sistema operativo, (apuntadores a las secciones de instrucciones y de datos, tablas de página, información de archivos abiertos, la pila, etc.), que sería la sección administrativa (Brawer, S, 1989).

Entonces se puede decir que un proceso es una abstracción de un programa corriendo (Tanenbaum, A.S.,1987). Una abstracción es una descripción de un sistema o parte de un sistema que no especifica todos los detalles completamente.

En un sistema operativo de tiempo compartido, conceptualmente cada proceso tiene su propia UPC. En realidad, por supuesto la UPC conmuta entre un proceso y otro. A este rápido cambio de procesos se le conoce como multiprogramación o multitareas.

Multiprogramación ó multitareas se emplea para optimizar la utilización de los recursos de la computadora. Uno de los principales recursos por optimizar sería la UPC, provocando que siempre tenga algo útil que ejecutar sin que se desperdicie su tiempo.

La ídea es la siguiente: el sistema operativo toma uno de los procesos que se encuentran en un recipiente de procesos y lo empieza a ejecutar. Eventualmente el proceso tiene que esperar para hacer algo, tal como esperar que un mando sea tecleado o que una operación de entrada-salida se realice. En un sistema de monoprogramación la UPC se desperdicia durante todo este tiempo. En un sistema de multiprogramación, el sistema operativo conmutará a otro

proceso y lo ejecuta. Cuando este proceso tenga que esperar, la UPC conmutará a otro proceso, y así sucesivamente. Eventualmente el primer proceso tendrá que dejar de esperar y obtendrá la UPC de nuevo. Así siempre habrá algún proceso que ejecutar y se evita que la UPC se desperdicie.

En la figura 3 inciso (a) vemos como se desarrolla en una computadora la multiprogramación de cuatro procesos en memoria. En el inciso (b) vemos como esto es abstraido dentro de cuatro procesos, cada uno con su flujo de control (su propio apuntador de programa) y cada uno corriendo independientemente de los otros. El inciso (c) muestra como se verian los procesos durante un intervalo de tiempo lo suficientemente largo. Todos los procesos han hecho progresos, pero en un instante de tiempo dado, solamente un proceso se encuentra corriendo.

Los sistemas operativos de multiprogramación deben mantener a todos los procesos en memoria principal con la finalidad de tenerlos listos para correrlos y conmutar entre ellos.

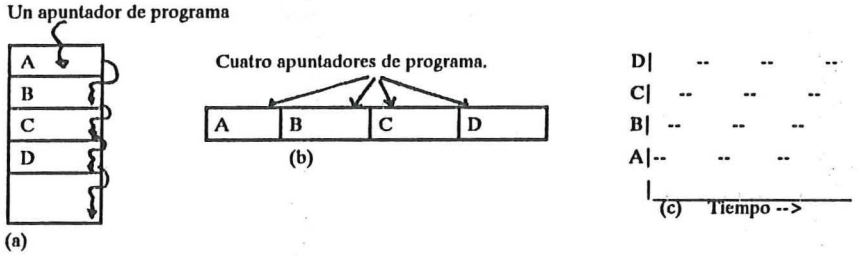

Figura 3.- (a) Multiprogramación de cuatro procesos. (b) Modelo conceptual de cuatro procesos secuenciales independientes. (c) Solamente un programa esta utilizando la UPC en un instante dado.

Para mantener varios programas en memoria al mismo tiempo se requiere de un hábil manejo de memoria, además si varios programas están listos para correr al mismo tiempo, algunas decisiones deben de tomarse para ello, administrando adecuadamente la UPC.

III.2.- Estados de los procesos.

Durante el progreso en la ejecución de un proceso, el proceso puede estar en un estado de un conjunto finito de posibles estados. Un esquema que clasifica los estados en que un proceso puede estar es el siguiente: nuevo, listo para correr, corriendo, esperando o bloqueado y muerto.

Estos estados son definidos de acuerdo a los estados transitorios por los que pasa el proceso. Por ejemplo la UPC puede ser compartida entre varios procesos y un proceso puede estar ya sea ejecutándose en la UPC o bien esperando a ser ejecutado por ésta. La figura 4 muestra los estados básicos posibles por los cuales pueden pasar los procesos.

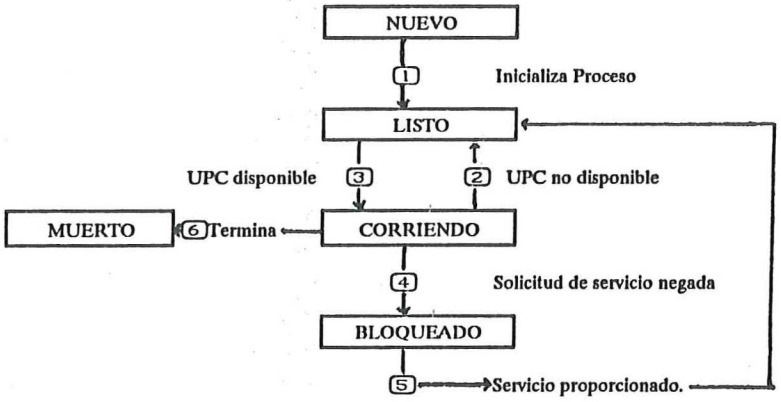

Figura 4.- Diagrama básico que muestra los estados por los que puede pasar un proceso. Mostrando las transiciones o funciones entre los estados.

El estado nuevo es la transición, entre un programa (entidad pasiva) y un proceso (entidad activa) que pasa al estado listo.

Un proceso que tiene todos los elementos para correr se identifica como un proceso que es ejecutable, pero si está temporalmente detenido esperando que se le asigne la UPC, dejando que otros procesos corran, se encuentra en el estado de listo.

El estado corriendo identifica a un proceso que se encuentra usando la UPC en un determinado instante.

Se dice que un proceso se encuentra en el estado bloqueado o suspendido cuando no tiene todos los elementos que necesita para correr y por lo tanto está en espera por alguna entrada-salida o bien que algun servicio le sea proporcionado para que pueda continuar.

El estado de muerto es el que identifica a un proceso que ha terminado.

Seis funciones o transiciones son posibles entre los estados de nuevo, corriendo, listo y bloqueado o suspendido. La figura No.4 muestra también las transiciones.

La transición 1 ocurre cuando un programa (entidad pasiva) pasa a ser un proceso (entidad activa). Las transiciones 2 y 3 son provocadas por el calendarizador de procesos (una política o estrategia del sistema operativo, que determina la manera de administrar el recurso). La transición 2 ocurre cuando se decide que el proceso que ha permanecido en el estado corriendo, ha tenido la atención de la UPC el tiempo suficiente y se le quita involuntariamente el control de la UPC. La transición 3 ocurre cuando se le asigna la atención de la UPC a un proceso que se encuentra en el estado de listo. La transición 4 ocurre cuando un proceso descubre que no puede continuar por que necesita algún elemento para estar listo para correr y tiene que esperar hasta que dicho elemento este disponible. La transición 5 ocurre cuando el elemento que necesitaba un proceso es proporcionado. La transición 6 ocurre cuando el proceso ha finalizado.

El término de calendarizado, esto es, la política de decidir cual proceso y por cuanto tiempo deba correr es un tema que se discutirá enseguida.

III.3.- Calendarizado de la unidad de procesamiento central (UPC).

A la política o estrategia de administración de los recursos que proporciona la computadora entre los procesos que se encuentran en ejecución se le llama calendarizado. Una adecuada calendarización de los recursos de la computadora es fundamental en un sistema operativo.

En un sistema de multitareas o multiprogramación la calendarización de la UPC es la base, para que mediante la conmutación de la UPC entre los procesos que están pasando de un estado listo al estado corriendo, se pueda utilizar con más eficiencia.

Un sistema operativo puede tener varias estrategias de calendarizado, dependiendo del recurso que se va a administrar. Para el calendarizado de la UPC exsisten dos principales estrategias. **El calendarizado a largo plazo y el de corto plazo.**

El calendarizado de largo plazo o calendarizador de tareas por colas (el término tarea es usualmete utilizado en sistemas por colas), es una estrategia de asignar durante mucho tiempo la atención de la UPC a un solo proceso. Los procesos se depositan en una cola del tipo primera que llegue, primera que se atiende, de la cual se extraen cuando se les asigna la UPC, pero un proceso no puede correr sino hasta que esté al frente de la cola.

Una vez que la UPC ha sido asignada a un proceso, este la retiene hasta que el mismo proceso indique su liberación. A ésta acción de liberar el recurso por el proceso mismo se le llama de **no prevaciado.**

El calendarizado de corto plazo o calendarizado de tiempo compartido, es una estrategia de asignar la atención de la UPC a los procesos de acuerdo a una tajada de tiempo. A cada proceso se le asigna su tajada de tiempo de atención de la UPC. Cuando se le agota su tiempo se procede a cambiar a otro proceso y así sucesivamente. A la acción de quitarle al proceso involuntariamente la atención de la UPC se le llama **prevaciado**

La principal diferencia entre estos calendarizados es la frecuencia de su ejecución. El de corto plazo debe seleccionar con más frecuencia a un nuevo proceso para que sea ejecutado o que continue ejecutándose, mientras que el de largo plazo los ejecuta con menos frecuencia.

En algunos sistemas el calendarizado de largo plazo puede eliminarse o ser mínimo.

Otro elemento involucrado en la calendarización de la UPC es el despachador. El despachador es un módulo que es el que proporciona el control de la UPC al proceso seleccionado por el calendarizador de corto plazo.

El módulo despachador realiza las actividades de salvaguardar y cargar los registros de los procesos (su contexto y su medio ambiente), conmutar la atención de la UPC a otro proceso, brincar a la dirección adecuada del proceso por ejecutar y regresar después de un tiempo, a la localidad exacta donde el proceso fue temporalmente detenido mientras que esperaba la atención de la UPC.

Se pueden tomar en cuenta diferentes criterios que determinaran a cual de los procesos que se encuentran en una cola asociada al estado de listos se le asignará la UPC.

El algoritmo de calendarizado "Round-robin" esta diseñado especialmente para sistemas que determinan como principal criterio, el compartir el tiempo de la UPC en forma equitativa entre los procesos. A estos sistemas se les conoce como sistemas de tiempo compartido. Se define en base a una pequeña cantidad de tiempo, llamada "Quantum" o tajada de tiempo. La tajada de tiempo es el período permitido para que un proceso retenga la atención de la UPC. Además al estado listo de los procesos se le es asignado una cola o lista de los procesos listos para correr y es tratada como una cola circular.

El módulo despachador de la UPC auxiliado por un apuntador, va señalando alrededor de la cola, asignando la UPC a cada proceso durante un intervalo de tiempo.

Para implantar el calendarizado de "Round-robin", la cola de listos es manejada como una cola en la que el primer proceso que llega es el primero que se atiende y los nuevos procesos son agregados al final de la cola de listos.

El despachador de la UPC recoge al primer proceso en la cola de listos y en el momento en que lo comienza a ejecutar, el módulo despachador determinará de acuerdo al cronómetro en tiempo real de la máquina, cuando interrumpir el proceso después de un intervalo de tiempo determinado.

Uno de dos posibles casos pueden presentarse: El proceso puede tener ocupado a la UPC un menor tiempo del asignado, en este caso, el proceso mismo relega la UPC en forma voluntaria, enviando una señal o bíen poniendo alguna marca (bandera) que indique que ha finalizado. Entonces se continuará con el siguiente proceso que se encuentra en la cola de listos.

El otro caso posible, es que el proceso consuma todo el tiempo asignado y además necesite más tiempo, entonces el tiempo se le terminará y provocará que el sistema operativo lo interrumpa.

Los registros así como el ambiente del proceso interrumpido son salvaguardados en una área de memoria exclusiva y segura y el proceso es colocado al final de la cola de listos.

El calendarizador de la UPC seleccciona el siguiente proceso (el primero) de la cola de listos y le asigna su tajada de tiempo durante el cual tendrá la atención de la UPC y así sucesivamente.

El funcionamiento del calendarizado "Round-robin" depende en gran manera de la tajada de tiempo asignada para cada proceso. En un extremo, si la tajada de tiempo es muy grande (infinito), el calendarizado Round-robin se comportará como un calendarizado de **no prevaciado,** en que el primer proceso que llegue será el primero que se atiende, hasta su terminación independientemente del tiempo que necesite para realizar todas sus funciones.

Si la tajada de tiempo es muy pequeña (digamos un milisegundo), el calendarizado Round-robin es llamado procesador compartido y parecerá (en teoría) como si cada uno de los "n" procesos tiene su propio procesador corriendo a 1/n la velocidad real del procesador.

Si hay "n" procesos en la cola de listos y la tajada de tiempo es "q", entonces cada proceso obtiene 1/n del tiempo de la UPC y a lo más "q" unidades de tiempo a la vez, despreciando el tiempo de conmutación. Cada proceso no debe esperar más que (n-1) x q unidades de tiempo hasta que le sea asignado de nuevo otra tajada de tiempo. Por ejemplo, si hay cinco procesos, con una tajada de tiempo de 20 milisegundos, entonces cada proceso podrá despertar hasta 20 milisegundos cada 100 milisegundos.

En programación, sin embargo, tenemos otro problema por solucionar, especialmente al final de la tajada de tiempo, en que recibimos una interrupción del temporizador.

El proceso que se lleva a cabo para que conmute la UPC a otro proceso, requiere de salvaguardar todos los registros y ambiente del proceso antigüo (el que acaba de utilizar su tajada de tiempo) y cargar todos los registros y ambiente del nuevo proceso. Este proceso es esencialmente el sobreencabezamiento de conmutación de contexto (salvaguardar y cargar todos los registros y ambiente) o tiempo muerto.

A el módulo que realiza la conmutación de contexto se le llama el despachador o conmutador de contexto (_asr).

Para ilustrar el efecto en tiempo de la conmutación de procesos en la eficiencia del calendarizado Round-robin, asumamos primero que tenemos solamente un proceso en ejecución y que tomará solamente 10 unidades de tiempo en finalizar su tarea.

Si la tajada es de 12 unidades de tiempo, el proceso terminará antes que se le agote la tajada de tiempo y por lo tanto no habrá tiempo muerto. Si ahora suponemos que la tajada de

tiempo es de 6 unidades de tiempo, el proceso necesitará dos tajadas de tiempo para realizar completamente su tarea, provocando una conmutación de contexto, haciendo más lenta la ejecución del proceso.

Ahora sí, la tajada de tiempo es de una unidad de tiempo, entonces se necesitarán nueve conmutaciones de contexto (nueve tiempos muertos) haciendo aun más lenta la ejecución del proceso. Entonces se requiere que la tajada de tiempo sea lo suficientemente largo con respecto al tiempo de conmutación de contexto y lo suficientemente pequeña con respecto al tiempo total de ejecución de los procesos.

Si la conmutación de contexto es de aproximadamente 10 por ciento el tiempo de la tajada de tiempo, entonces alrededor del 10 por ciento del tiempo de la UPC será gastado en tiempo de conmutación de contexto.

III.4.- Manejo de memoria.

En multitareas o multiprogramación, la UPC se conmuta rápidamente entre uno y otro proceso, aumentando la velocidad en respuesta de la computadora hacia los procesos. Para realizar este rápido cambio de contexto, los procesos deben estar en memoria principal. Por lo tanto debemos asegurar que cuando un proceso este corriendo, accese la región de memoria correcta. Cada proceso debe tener una porción de memoria privada, así como también puede haber localidades de memoria que sean accesadas por todos los procesos.

A todas aquellas localidades de memoria que pueden ser accesadas por todos los procesos se les llama memoria compartida. A todas aquellas localidades de memoria que pueden ser accesadas por un solo proceso es memoria privada.

En multitareas o multiprogramación son posibles dos esquemas de manejo de memoria. Cada uno de los esquemas divide la memoria en un número de regiones o particiones.

Estos esquemas son: La asignacíón de múltiples particiones de memoria contiguas fijas (MPF) y se conoce como multiprogramación con un número fijo de procesos, el otro esquema es la asignación de particiones múltiples de memoria contiguas variables (MPV) y se llama multiprogramación con un número variable de procesos.

La necesidad de interrumpir espontaneamente un proceso para realizar la conmutación de contexto, forza a una organización particular: Los programas deben ser considerados como una serie separada de componentes que puedan ejecutarse concurrentemente. Los componentes son los procesos; la organización es llamada multitareas o multiprogramación. Cada uno de los procesos constituye un programa completo que es capaz de ejecutarse en forma independiente.

Cada proceso tiene su dirección de código que ejecuta, su propia pila y su propia área de datos local. Estas son áreas de memoria dedicadas, en los cuales el proceso puede tener parámetros por llamadas a procedimientos, direcciones de retorno, datos temporales y variables similares que no son compartidas con otros procesos. Además cada proceso tiene su propio conjunto de valores para el contador de programa y el apuntador de pila, más cualquier otro registro general, especial y/o coprocesador que la circuitería proporcione.

III.5.- Condiciones de carreras.

La situación de condiciones de carreras se presenta cuando el resultado conjunto de dos o mas procesos depende de la velocidad relativa con la cual los procesos se ejecutan.

Suponiendo que dos procesos están cooperativamente tratando de obtener una meta: el proceso "0" y el proceso "1"; sin embargo para llegar a esa meta el proceso "0" debe terminar de realizar cierta parte de su código para que el proceso "1" pueda continuar y realizar su propia parte en forma exitosa. Si el proceso "1" se adelanta debido a una mayor velocidad relativa se obtendrá un resultado incorrecto.

En general si tenemos varios procesos contribuyendo para obtener una meta común que depende de las velocidades relativas de cada uno de ellos, se necesita de un mecanismo de sincronía para forzar a todos los procesos a esperar mientras que terminan los procesos que se requiere finalizar primero.

A este mecanismo se le llama Barrera. La barrera es un mecanismo de sincronización entre los procesos y actua como sigue.

La barrera es una estructura que tiene asociado por lo menos un arreglo de enteros inicializado en un cierto valor (número de bloqueo) que sirve como contador y colas múltiples asociadas a un estado de espera. Cuando un proceso llega a la barrera verifica cuantos procesos más han llegado antes. Si el número de procesos incluyendo al que acaba de llegar es menor que el número de bloqueo, entonces el proceso recien llegado tiene que esperar, disminuyendose el valor del número de bloqueo. Por otro lado si el número de procesos que se encuentran en la barrera, más el que acaba de llegar es igual al número de bloqueo, entonces todos los procesos que estaban esperando, incluyendo al último que llegó a la barrera se les permite proceder y se reinicializa a su valor original el número de bloqueo (Brawer, S., 1989).

III.6.- Secciones críticas.

Una sección crítica es un conjunto de declaraciones o datos que solo pueden ser procesados por un proceso a la vez. En un sistema de multitareas se presenta la posibilidad de que varios procesos necesiten compartir al mismo tiempo algun área de datos o de instrucciones.

Un mecanismo que evite los probleams que involucran compartir memoria, archivos, códigos de instrucciones etc., es la implantación de exclusión mutua.

Peterson, J. L., 1985, y Tanenbaum A. S 1987, presentan varios algoritmos para solucionar el problema de la sección crítica.

III.7.- Semáforos.

Para resolver los problemas que se presentan en la implantación de exclusión mutua, una herramienta de sincronización llamada semáforo fue presentada por Dijkstra 1965a.

Un semáforo esta formado por una estructura que contiene un valor entero y un apuntador a una lista o cola de procesos. El semáforo aparte de su inicialización puede proporcionar acceso solamente a través de dos operaciones atómicas: "P" y "V", cuyas definiciones son las siguientes.

La operación "P" sobre el semáforo S, P(S) opera como sigue:

```
struct semáforo {
        int S;
        cola *apuntador;
        }semáforo;

        P(S): S = S - 1 ;
        Si S < 0;
        {
        agregar este proceso a una cola de espera;
        duerme();
        SI NO
        pasa y continua;
        }
```

La operación "V" sobre el semáforo S, V(S), opera como sigue:

```
        V(S): S = S + 1;
        Si S < = 0;
        {
        tomar un proceso de la cola;
        despierta ();
        pasa y continua;
        }
```

El proceso que hace una operación P(S) decrementa el valor del entero del semáforo, si el valor es menor que 0 lo pone en un estado de espera agregándolo a la cola, si no, pasa y continua haciendo su trabajo y así para los demás procesos que hacen un acceso al semáforo.

Eventualmente algun proceso que pasó y continuó a través del semáforo, realiza una operación V(S) para permitir que otros procesos que esten en la cola del semáforo puedan continuar.

En la operación V(S), el proceso que hace esta operación primero incrementa el valor del entero del semáforo, si el semáforo es menor o igual a 0, toma el primer proceso que está en la cola de espera del semáforo y lo despierta para continuar con su trabajo. Si el semáforo es un valor negativo, su magnitud indica el número de procesos que esperan en la cola del semáforo. (Tsichritzis y Bernstein, 1974).

Las modificaciones al valor entero del semáforo, la comparación y la acción de colocar o tomar un proceso de la cola del semáforo en las operaciones "P" y "V" deben ser ejecutadas indivisiblemente. Esto es, que cuando un proceso este realizando un acceso a un P(S) o V(S), ningún otro proceso puede simultáneamente estar modificando al valor del mismo semáforo.

III.8.- Abrazo mortal.

Un conjunto de procesos se encuentran en estado de abrazo mortal, cuando cada proceso del conjunto, está esperando por algún evento que solo puede ser causado por otro proceso del conjunto (Peterson, J. L.,1985).

En un sistema de multitareas o multiprogramación, varios procesos pueden competir por un número finito de recursos (un recurso puede ser un dispositivo de circuitería o una pieza de información), provocando que en cierto momento un proceso se quede en un estado de espera para siempre cuando hace la petición de un recurso el cual nunca será proporcionado.

Para que suceda un abrazo mortal se deben presentar cuatro condiciones necesarias:

Exclusión mutua. Cada uno de los recursos, es ya sea, asignado o disponible a exactamente un solo proceso.

Proceso detenido y esperando. Debe existir un proceso que este reteniendo el uso de algún recurso y estar esperando por un recurso adicional.

Que no se pueda quitar el recurso a un proceso en forma involuntaria (no prevaciado). El recurso puede solamente ser liberado voluntariamente por el proceso.

Condición de espera circular. Que exista una cola circular de dos o más procesos, cada uno de los cuales esta esperando por un recurso que es retenido por el siguiente miembro de la cola.

Para prevenir la situación de abrazo mortal, debemos asegurarnos que al menos alguna de éstas condiciones nunca se presente.

Hay tres métodos básicos para prevenir el abrazo mortal.

Antes de proceder con la exclusión mutua:

a) Cada proceso debe obtener todos los recursos que necesita y se evita que este detenido y esperando.

b) Si un proceso está reteniendo algun recurso y solicita otro y éste no se le asigna inmediatamente, entonces el proceso debe liberar el recurso que está reteniendo. Se le quita el recurso en forma involuntaria (prevaciado).

c) Imponer una ordenación lineal sobre todos los tipos de recursos. Los procesos pueden solicitar recursos cuando lo deseen, pero las peticiones deben ser hechas en orden númerico. Un proceso puede pedir primero un manejador de disco y luego la impresora, pero no puede pedir primero la impresora y luego el manejador de disco. Eliminando de esta manera condición de espera circular.

III.9.- Código reentrante.

En multitareas, especialmente en sistemas de tiempo compartido, es típico que muchos procesos necesiten estar ejecutando programas comunes. Si se les proporcionara copias individuales de estos programas a cada proceso, entonces gran parte de almacenamiento

primario podria ser desperdiciado. La solución es compartir estos programas o partes de código. El compartir estos códigos debe ser cuidadosamente controlado para prevenir que un proceso este modificando la información mientras que otro proceso la está leyendo.

El código que no se modifica mientras esta en uso se dice que es **reentrante.** Si el código es reentrante entonces éste nunca cambia durante su ejecución. Así dos o más procesos pueden ejecutar el mismo código "al mismo tiempo".

El código que puede ser utilizado pero que debe ser reinicializado cada vez que se usa se dice que es **"serialmente reusable".**

El código reentrante puede ser compartido por varios procesos a la vez mientras que el serialmente reusable puede ser solamente usado por un proceso a la vez, (Deitel, H. M.,1984).

El sistema operativo MS-DOS es del tipo de código serialmente reusable o código no reentrante. De aquí que se necesite considerar al código del MS-DOS como sección crítica y de exclusión mutua.

III.10.- Comunicación interprocesos.

En un sistema de multitareas es de gran utilidad poder comunicar los procesos entre si. Los mecanismos de comunicación incluyen: semáforos, ductos o pipas, colas, mensages, señales, pipas nombradas, memoria compartida y sockets.

El semáforo ya se describió anteriormente.

Un ducto o pipa es un mecanismo que permite que un flujo unidireccional de información pase de un proceso padre a un proceso hijo. Los datos siempre fluyen desde el proceso padre que escribe en un extremo hacia el proceso hijo que lee en el otro extremo. Las pipas son frecuentemente usadas para proporcionar la salida de una implantación dirigida a otra como entrada y generalmente se implantan usando el sistema de archivos.

Los mensajes son bloques de datos con un formato definido por el sistema o por la aplicación involucrada. Los formatos exactos y semántica de los mensages varía ampliamente de sistema a sistema.

Una señal (algunas veces llamada interrupción por programación) le indica a un proceso que suspenda lo que esta haciendo e inmediatamente maneje un evento asíncrono. Usualmente, esto es realizado por la ejecución de una pieza de código que se invoca al recibir la señal. Al finalizar con la atención de la señal asíncrona, el sistema debe ser capaz de continuar con lo que estaba haciendo, justo en el momento en que fue interrumpido.

Las pipas nombradas son similares a los ductos o pipas en que ambas permiten intercambio de datos entre los procesos de un carácter a la vez. A diferencia de las pipas ordinarias, estas pueden soportar la trasmisión de largos bloques de datos y llevar la información en ambas direcciones (tal como lo haría un par de pipas ordinarias) y pueden ser entre procesos que no tengan relación de parentesco. Pueden existir durante más tiempo que el (los) proceso (s) que la(s) crearon. El acceso a ellos es como un archivo ordinario y existen dentro del sistema de archivos.

La mayoria de los sistemas de multitareas proporcionan una forma para que dos o más procesos compartan un bloque de memoria común y tal vez esta forma de comunicación interprocesos sea la más rápida. En **memoria compartida** otros mecanismos (tales como semáforos) se ven con frecuencia para arbitrear el acceso al área común.

La versión de UNIX desarrollada por la Universidad Berkley en California contiene un mecanismo de comunicación interprocesos llamado **socket**. Los sockets permiten que un proceso sobre una red de computadoras inicie un programa dentro de otro, proporcione entradas al programa y reciba la salidas del programa.

III.11.- Sistemas de tiempo real.

Una propiedad fundamental de los sistemas o programas en tiempo real, es que una o todas sus entradas llegan del mundo exterior de manera asíncrona con respecto a cualquier trabajo que el sistema este haciendo.

El sistema debe ser capaz de interrumpir inmediatamente su actividad actual y ejecutar algun código predefinido que capture o responda a una entrada-salida, la cual es frecuentemente una señal transitoria o fugaz. La captura del nuevo dato en su momento, podría disparar la ejecución de uno o más programas urgentes que estuvieran esperando la entrada-salida. Finalmente, el sistema debe ser capaz de reanudar sus actividades originales (Ripps, D. L., 1990).

La necesidad de interrumpir espontaneamente una parte del sistema para realizar alguna tarea, forza a una organización particular. Las aplicaciones en tiempo real deben ser escritas como una serie separada de componentes de programas que puedan ejecutarse concurrentemente. Los componentes son llamados tareas o procesos.

Cada una de los procesos es un programa completo que es capaz de ejecutarse en forma independiente. Cada proceso tiene un segmento de código a ejecutar, tiene su propia pila y su propia área de datos local.

El sistema de tiempo real opera bajo el control de un código residente (sistema operativo), el cual decide, que, como y cuando ejecutar un proceso.

III.12.- Multitareas.

En resumen; multitareas es la capacidad de una computadora (con un solo procesador) de correr uno o más programas a la vez, de tal manera que, parezca ante el usuario que los

programas están corriendo simultáneamente. Cada programa opera como si tuviera una computadora para él mismo, olvidándose de los otros programas que están corriendo y del mismo sistema operativo de multitareas.

El recurso clave que un sistema de multitareas debe manejar es la UPC. Aunque hay muchos esquemas que pueden ser usados para distribuir el tiempo de la UPC entre los diferentes procesos, todos pueden ser ubicados en dos categorias: El esquema de **prevaciado** y el esquema de **no prevaciado.**

Aunque el calendarizado de no prevaciado es relativamente más fácil de implantar, éste generalmente no es tan tolerante a errores como lo es el de prevaciado. Por ejemplo en un calendarizado de no prevaciado, un proceso que falla al abandonar la UPC, debido probablemente a un problema por programación o por circuitería, detendrá al sistema por completo, destruyendo todos los datos utilizados por los otros procesos que cooperaban para realizar una tarea en común.

Además de compartir el tiempo de la UPC, todos los sistemas de multitareas deben proporcionar memoria compartida entre las aplicaciones. Las tres funciones principales que un sistema puede proporcionar en la administración de memoria incluyen; asignación de almacenamiento, conmutación de almacenamiento ó paginación y protección.

En un sistema de multitareas, cada tarea debe ser confinada a una específica parte de la memoria y no debe de intentar manipular memoria que le corresponda a otra sin permiso previo.

Para la conmutación de programas de memoria principal a disco, los procesos deben ser detenidos, ponerlos en disco y realizar la acción inversa cuando haya suficiente espacio de memoria principal.

Para todos aquellos procesos en que el espacio de memoria convencional es insuficiente, se requiere de una unidad de manejo de memoria.

Figura 5.- Multitareas, conmutándose de un proceso a otro haciendo transparente las implantaciones.

Una unidad de manejo de memoria facilita la conmutación de memoria principal a disco y viceversa, detectando automáticamente y de manera rápida cuando un programa trata de obtener información que no esta en memoria principal. Cuando detecta un acceso como tal, ésta activa rutinas de atención que leen la información necesaria del disco y entonces continuan con el programa donde fue interrumpido. La figura 5 muestra un abstracción del concepto de multitareas.

IV.- COMUNICACION EN SERIE ASINCRONA.

El elemento de comunicación, el receptor trasmisor universal asíncrono (UART) 8250 de INTEL es diseñado para enviar y recibir datos en serie y en forma asíncrona.

La comunicación de información digital se puede realizar en paquetes discretos de datos que se envian y se reciben, teniendo la oportunidad de contar con dos opciones.

Podemos determinar que un paquete sea de un solo carácter (transmisión paralela) o que sean paquetes de un solo bit (transmisión serie). Si enviamos paquetes de carácter, debemos tener una línea de datos por bit u ocho líneas de datos para un carácter ASCII extendido. Además si deseamos enviarlo en ambas direcciones al mismo tiempo, necesitamos dieciseis lineas, sin mencionar las líneas de control. Sin embargo el desarrollo de toda una red de comunicación con éstos requerimientos resulta ser sumamente cara. Una alternativa que puede simplificar los requerimientos de alambrado podria ser el uso de la red telefónica existente para trasmitir y recibir datos de computadora a computadora. La desventaja es que este medio de transmisión es completamente análoga y el esquema de transferencia de datos es en serie.

La transmisión en serie provee tazas (cantidad de información) de transmisión más bajas que la transmisión paralela. Sin embargo existen aspectos que equilibran o compensan los aspectos negativos. Para microcomputadoras podría prescindirse de razones de transmisión extremadamente altas. Así que podemos realizar la mayor parte de la comunicación via transmisión en serie.

La diferencia entre sistemas de transmisión asíncrona y síncrona es la existencia o inexistencia de un subsistema de control de señalización que sincronice la transmisión de datos.

En un sistema de transmisión síncrono, existe una señal separada que sincroniza la transmisión y recepción. A diferencia en un sistema de transmisión asíncrona no existe tal mecanismo de sincronía. La sincronización se obtiene por medios diferentes.

Para la transmisión asíncrona en serie, el mecanismo de sincronización es el bit de inicio que precede a cada flujo de bits de datos. La figura 5 muestra el formato de dato via transmisión en serie.

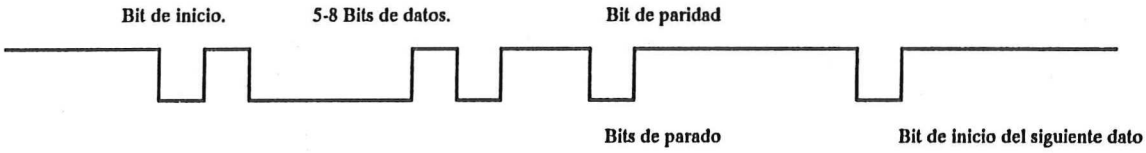

Figura 6.- Formato de dato via trasmisión serie.

El bit de inicio permite al UART sincronizar su reloj interno con los datos que van llegando. De aquí que, éste utiliza su reloj interno para muestrear las señales en aproximadamente a la mitad del tiempo del bit que esta llegando y recorrer el valor muestreado (uno o cero) metiéndolo en un extremo de un registro de recorrimiento. Cuando llegue(n) el (los) bit(s) de parado, que indica(n) el fin del carácter, entonces el UART mueve el carácter ensamblado dentro de un espacio de memoria reservado y le indica a la UPC que el carácter ha sido recibido.

El UART esta diseñado para permanecer en contacto con el transmisor solamente hasta el final de un solo carácter, así que se estará autosincronizando para cada carácter que recibe. La característica de la transmisión asíncrona es el hecho de que los datos pueden arrivar en cualquier momento.

Por acuerdo, la linea de comunicación entre datos es mantenida en alto (1), llamada condición de marca. La condición de espacio es el término aplicado cuando la linea esta en bajo (0). Un bit de inicio es definido como la señal que mantiene en 0 (espacio) a la linea de comunicación durante el tiempo de un bit. Cualesquier tiempo más pequeño es interpretado como ruido. El (los) bit (s) de parado están definidos como la señal que mantiene en alto (1)

a la señal de comunicación durante el tiempo de uno, uno y medio o dos bits, dependiendo de como es inicializado el UART. Tanto el receptor como el trasmisor deben ser inicializados con el mismo número de bits de parado.

En general la comunicación apropiada entre un receptor y un trasmisor requiere que ambos sean inicializados para aceptar el mismo número de bits de datos (5, 6, 7 ó 8), la misma convención de paridad (paridad par, inpar ó ninguna), la misma tasa de señalización (baud rate) y la misma longitud de bits de parado (Biggerstaff, T. J., 1985).

V.- DISEÑO DEL SISTEMA DE MULTITAREAS.

El diseño es un proceso creativo. No existen reglas exactas que se deban seguir para diseñar un sistema operativo, uno mas bien debe aprender de su propia experiencia o de la experiencia de otros y entonces decidir cuales técnicas contribuyen a un buen sistema y cuales problemas deben ser evitados. Sin embargo, hay una guia general y sobre todo aproximaciones que pueden seguirse.

Existen básicamente tres aproximaciones como técnicas de diseño: La metodología de diseño de arriba-abajo, la técnica de diseño estructurada y la técnica de diseño orientada a objetos (Tsichritzis y Bernstein 1974),(Savitzky 1985).

El diseño del sistema operativo que se desarrolló en este trabajo se enfoca principalmente a un diseño estructurado. Esta metodología de diseño usualmente implica el uso de estructuras para separar las partes del sistema, de tal manera que cada parte puede diseñarse por separado.

El diseño consistió en lograr un sistema de multitareas en tiempo real en modo texto para la computadora personal IBM y 100% compatibles, que se ejecuta en la cima del MS-DOS versión 2.0 y más recientes, con un ambiente de ventanas como interfaz con el usuario.

Para lograrlo se estableció como mecanismo principal la administración del tiempo de la UPC en forma equitativa entre los procesos. Se escogió el calendarizado Round-robin, que proporciona la capacidad de realizar un sistema de prevaciado. Se estableció una tajada de tiempo como mínimo de 50 milisegundos, tiempo aproximado entre una y la siguiente señal del cronómetro en tiempo real (18.2 veces por segundo). Como máximo un tiempo de segundo y medio, que se determinó en base al tiempo de respuesta del MS-DOS cuando actua sobre el manejador de disco flexible.

Todos los procesos tienen espacio de memoria privada donde tienen que trabajar y para evitar el acceso otras localidades de memoria que no sean las consideradas como memoria compartida.

A cada proceso se le es asignado un nodo o celda en una cola implantada en una lista ligada circular, que constituye su tabla de control.

El cambio de contexto de los procesos se realiza por un módulo despachador (_asr) y un módulo auxiliar (int 85). La conmutación de procesos se realiza de acuerdo a la tajada de tiempo asignada. Sin embargo el sistema conmuta a otro proceso si éste no usa su tajada de tiempo, sea por estar en espera de alguna operación de entrada-salida o por que necesitó menos del tiempo, cediendo la tajada de tiempo o parte de está al siguiente proceso por conmutar.

V.1.- Estructuras de datos.

Se utilizó el lenguaje turbo "C" para la formación de las diferentes estructuras de datos, apuntadores, variables, parámetros y funciones.

Las tablas de control de cada uno de los procesos se encuentran en una lista ligada asociada al estado listo de los procesos. Estas consisten de varias estructuras de datos con los siguientes parámetros por los que se rige cada proceso.

- Una estructura del tipo definido imagen, que contiene todos los registros de circuitería del procesador que representan el contexto del proceso, donde se mantiene y se salvaguarda su estado.

```
typedef struct{
        unsigned int bp,di,si,ds,es,dx,cx,bx,ax,ip,cs,flags;
                }imagen;
```

- Una estructura del tipo definido cola que contiene un conjunto de parámetros que definen la tabla de control de cada uno de los procesos:

```
typedef struct cola {
        imagen far *pilita;
        struct cola *sig;
        unsigned int estado;
        unsigned int tarea_numero;
        unsigned char far *espacio_teclado;
        unsigned char cursor_renglon;
        unsigned char cursor_columna;
        unsigned char cursor_der_limite;
        unsigned char cursor_izq_limite;
        unsigned int ss_peligro;
        unsigned int sp_peligro;
        unsigned int cuenta_peligro;
        unsigned int curr_wnd;
        unsigned char cursor_sup_limite;
        unsigned char cursor_inf_limite;
        int izquierda;
        int cima;
        int derecha;
        int fondo
        int adelante;
        int atras;
        int marco;
        int sombra
        unsigned char atributo;
        unsigned char visible;
                }COLA;
```

Cada uno de los parámetos representa lo siguiente:

- Un apuntador que indica la dirección (segmento de pila SS y apuntador de pila SP) donde se encuentra la pila o área de memoria de trabajo del proceso, (imagen far *pilita;).
- Un apuntador que señala la siguiente tabla de control (celda) encadenada en la cola, (struct cola *sig;).
- Un valor entero que indica el estado actual del proceso. Si es 1 está en estado activo, esto es, que el proceso está pasando de un estado listo al estado corriendo respectivamente. Un 0 si está congelado o dormido hasta que el usuario le indique que se active de nuevo, pero permanecerá muerto (con valor cero) si violó alguna de las restricciones del sistema, (unsigned int estado;).
- Un valor entero que indica el número de proceso asignado a la tabla de control, (unsigned int tarea_numero;).
- Un vector que indica la dirección del espacio de memoria reservado para el teclado, correspondiente al proceso en particular, (unsigned char far *espacio_teclado).

- Un par de valores, renglón y columna, que indican la posición del cursor para un proceso en específico, (unsigned char cursor_renglon, unsigned char cursor_columna;).
- Dos pares de valores que indican la posición de los límites izquierdo, derecho, del desplazamiento horizontal y la posición de los limites superior e inferior del desplazamiento vertical que puede realizar el cursor para un proceso en específico, (unsigned char cursor_izq_limite, cursor_der_limite, cursor_sup_limite, cursor_inf_limite;).
- Un par de enteros que salvaguardan los valores de los registros de segmento de pila (SS) y de apuntador de pila (SP), de la pila de trabajo del proceso que logró entrar a la sección crítica (sección no reentrante). Estos valores nos permiten regresar a la misma pila de trabajo cuando se pierde el control al interrumpir al MS-DOS por exceder su tiempo de respuesta, (unsigned int ss_peligro,sp_peligro;).
- Un entero que mantiene el valor del contador que se inicializa con cierto valor (el valor de la variable reloj +250) al presentarse un problema con algún periférico, dando tiempo para desplegar un mensaje de error y permitir al usuario la recuperación del periférico, (unsigned int cuenta_peligro).
- Nueve valores que proporcionan las características del área de pantalla (ventana) reservada para un proceso en particular. Describiendo la esquina superior izquierda (int izquierda, cima;), la esquina inferior derecha (int derecha, fondo;), los atributos de caracteres y respaldo (int adelante, atras;) y si van a llevar marco y proyectar sombra (int marco, sombra;).
- Un valor que indica si la ventana del proceso se va a visualizar en pantalla o estará oculta. Si el valor es 1, se visualiza, si es 0, se oculta, (unsigned char visible;).

Para implantar el calendarizado Round-robin, es necesario crear un apuntador que continuamente este indicando la tabla de control del proceso que se va a ejecutar, e ir moviéndolo para poder indicar cual es la siguiente tabla de control del proceso que se va a apoderar del control de la UPC, cada vez que se termine la tajada de tiempo asignada para cada proceso.

Se creó un apuntador (COLA *cabeza;) que realiza ésta función, señalando a las tablas de control que se encuentran circularmente ligadas y que se va recorriendo cada vez que se termina la tajada de tiempo de la siguiente tabla de control. Además se presenta la necesidad de leer o actualizar los valores que están contenidos en las tablas de control.

Ya que el apuntador *cabeza continuamente esta en movimiento se creó otro apuntador (COLA *cabeza_movil;) que es utilizado para hacer lecturas y modificaciones a los parámetros de las tablas de control.

Para tener un punto de referencia del origen de la cola, que esté apuntando siempre a la primera tabla de control, se creó el apuntador (COLA *cabeza_inicio;).

Todo aquel programa que se desee ejecutar en el sistema de multitareas, debe ser compilado en conjunto con el sistema mismo. Teniendo así la información de la sección administrativa, sección de datos y sección de código del proceso. A cada proceso se le asigna su correspondiente tabla de control y pila de trabajo privada.

Al inicio del sistema, despúes de la formación de la tabla de control del procesador de mandos (proceso número cero) y del primer proceso, se indica el número de programas que se han cargado menos uno. **Esta información se le da en el programa.** De acuerdo a este número, se procede a generar espacio para cada una de la tablas de control correspondiente a cada programa.

Cada una de las tablas de control es cargada con la información inicial en el momento en que es inicializado el sistema. Para ésto se empleó un apuntador temporal (COLA *temp;) que apunta al inicio de la cola. Con este apuntador temporal, se procede a llenar con los parámetros iniciales las tablas de control con los que se empezaran a ejecutar cada uno de los procesos.

Debido a que la función "malloc" de "C" reserva espacio hacia la parte alta de la memoria reservada para el usuario, a partir de la última dirección que él mismo reservó, se procede a generar un espacio lo más grande posible mediante "malloc" para ubicar a las áreas de trabajo de cada proceso en la parte más baja de la memoria posible. A partir de aquí se les va asignando espacio de memoria privada para cada uno de los procesos.

Despues de que se terminó de cargar cada una de las tablas de control, una para cada proceso y de colocar las áreas de trabajo en lo más bajo de la memoria posible, se procede a liberar el espacio restante.

Una función de inicialización del trasmisor receptor asíncrono (UART) es incluida en lenguaje ensamblador para establecer comunicación mediante el puerto serie via interrupciones. La interrupción reservada (int 0BH para el COM1 o COM2) para comunicación via puerto serie es intervenida para que apunte a una rutina "void recibe_datos (void)" que esta encargada de recibir datos y ponerlos en un arreglo de 500 elementos (unsigned char dato_uart[500];).

V.2.- Núcleo del sistema de multitareas.

Hasta aqui todo fue una preparación especial para que los programas puedan ser ejecutadas en forma simultanea en el ambiente de multitareas. Es en este momento en que la información de cada una de las tablas de control estan completadas, los procesos tienen su propio espacio de memoria privado de trabajo y los programas compilados (procesos) listos para proceder a su ejecución.

Se invoca al programa principal (inicio ()) que representa el núcleo del sistema de multitareas en tiempo real. La mayor parte del núcleo es codificado en lenguaje de bajo nivel "Turbo ensamblador".

Lo primero que se procede a hacer es salvaguardar la pila de trabajo donde realiza las actividades "Turbo C", para que posteriormente al finalizar multitareas sea el mismo "Turbo C" quien reestablezca todos los parámetros que él mismo modificó.

Todos los accesos a los servicios del sistema se realizan mediante interrupciones, de aqui que se procede a capturar un conjunto de vectores de interrupción, para hacerlos que apunten a nuestro propio código y manipular el proceso de acuerdo a las necesidades que presenta el

ambiente de multitareas. Se crearon otros vectores que apuntan a funciones que ayudan al buen funcionamiento del ambiente reentrante de multitareas. La tabla VIII describe los vectores de interrupción que se interceptaron.

Tabla VIII.-Vectores de interrupción que se interceptaron y redirigieron.

Int.original	Capturada por:	Cambiada a:
int 08H	_asr (módulo despachador)	int 88H
int 09H	_cap09 (manejador de teclado)	int 89H
int 10H	_cap10 (manejador de video)	int 80H
int 1bH	_uno_bh (conmuta a procesador de mandos)	_dirección_1bh
int 21H	_cap21(manejador de las funciones de la interrupción 21H del MS DOS)	int 81h
int 24H	_cap24 (manejador de errores)	_dirección_24h
int 0bH	_recibe_datos (receptor de datos via puerto serie)	_dirección_0bh

De los nuevos vectores de interrupción (Tabla IX) estan:

Tabla IX.-Nuevos vectores de interrupción.

int 82h (Saca foto al utilizar periféricos lentos)
int 83h (Quita foto al utilizar periféricos lentos)
int 84h (Recuperador de procesos por exceso de tiempo)
int 85h (Módulo auxiliar).

El vector de interrupción correspondiente a la interrupción 08H dirección, que apunta a la rutina de atención de la hora del día y que se dispara en aproximadamente 18.2 veces por segundo, proporciona una señalización en tiempo real.

La conmutación de contexto se rige mediante la señalización que envia el circuito temporizador (8254) a través del canal 0, hacia la linea IRQ0 del circuito controlador de interrupciones (8259-A). Este a su vez le indica al microprocesador (8088/8086) que se detectó esta interrupción, el cual a su vez invoca la interrupción **INT 08H del BIOS. Esta interrupción fue capturada para que cada vez que se invoque le llegue al módulo despachador (_asr) de procesos, que es el asignador de recursos y conmutador de contexto (int 08H -> _asr (módulo despachador)).**

El módulo despachador salvaguarda todos los registros en el momento de su llegada, además invoca a la interrupción original del reloj (ahora int 88H) para que realice las actividades normales destinadas al reloj. Por ejemplo incrementar un contador de 4 bytes de la hora del día en la dirección 0040:006CH que se encuentra en el área de datos del BIOS (para la IBM PC XT).

Antes de realizar la conmutación de procesos, el módulo despachador verifica si existe algún proceso dentro de sección crítica (código del MS-DOS), mediante la consulta a un semáforo llamado candado. Cuando candado esta en 1 implica que el candado esta quitado, por consiguiente no hay proceso que se encuentre en sección crítica y procede a hacer la conmutación de procesos. Si candado es 0, el candado esta puesto señalando que hay algún proceso dentro de sección crítica.

En el caso en que el semáforo este puesto (1) el módulo despachador pone una señal (bandera) indicando que un proceso desea entrar a la sección crítica. Incrementa una variable (cuenta_tajadas) que cuantifica el tiempo que permanece algun proceso dentro de sección crítica. Si excede de cierto tiempo (aprox. un minuto y medio) el proceso que se encuentra en la sección crítica es interrumpido, se restablece todo su contexto y ambiente a la posición exacta antes de entrar a sección critica, permitiendo enseguida que otros procesos puedan entrar a la sección crítica.

La interrupción int 84H es la encargada de restablecer el contexto y ambiente de los procesos por exceso de tiempo dentro de sección crítica. Este exceso de tiempo generalmente se presenta por problemas que se registran en algún periférico, el cual tarda tiempo en contestar para continuar la ejecución en tiempo real. **Durante la atención del sistema hacia un periférico lento el sistema de tiempo real permanece muerto durante un minuto y medio.** Sin embargo para usos más precisos este tiempo puede ser modificado (actualmente) en el programa fuente.

En el caso en que el semáforo este quitado (0) y sea el inicio (arranque = 0), el módulo despachador procede a realizar la conmutación de procesos. Inicializa las variables que indican que aun nadie desea entrar a la sección crítica (bandera = 0) y al contador (cuenta_tajadas) que indica el tiempo que dura un proceso dentro de sección crítica. LLama a la función

(posiciona_cursor) que extrae de la tabla de control los valores de la posición del cursor. Posiciona el cursor a través de la interrupción de servicio a la pantalla (int 80h) en el área adecuada (ventana) para cada uno de los procesos que necesiten desplegar información en la pantalla.

Si el proceso visualizado tiene asignado el recurso de teclado (_tarea_actual = _tarea_permitida), el módulo despachador coloca el apuntador del cursor "> >", en la ventana del proceso, de lo contrario el cursor estará oculto.

Si no es el inicio (arranque = 1) el módulo despachador llama primero a la función que realiza el respaldo de la posición del cursor (salva_cursor) dentro de la tabla de control correspondiente al proceso visualizado y procede con la conmutación de procesos.

Al salir, el módulo despachador reestablece los valores del estado de todos los registros de la UPC tal como se encontraban.

De las interrupciones que se crearon se encuentra **el módulo auxiliar (int 85H). Este es utilizado para realizar el cambio de contexto y quitar el candado desde dentro de la función que intercepta a la interrupción 21H (_cap21) del MS-DOS.** Este módulo auxiliar realiza lo mismo que el módulo despachador, pero éste a diferencia del módulo despachador que se rige de acuerdo a la señalización que proviene del temporizador, es llamado desde módulos por programación.

La asignación del recurso de teclado es responsabilidad del usuario, que es el encargado de determinar mediante un mando, con cual de los procesos desea interactuar. El mando que le permite realizar esta elección es la interrupción reservada para el usuario Ctrl-Break que **es interceptada a través de la función (_uno_bH).** La tarea de esta interrupción es hacer que la variable _tarea_permitida sea igual a 0. Proporcionando al usuario conmutar e interactuar con el un procesador de mandos que se describirá más adelante.

De las interrupciones que se interceptaron para su manipulación y correcto funcionamiento dentro de un ambiente de multitareas se encuentra la **interrupción 21H del MS-DOS (_cap21).**

Esta función tiene como tarea entre otras, verificar si alguno de los procesos que se encuentran en ejecución intenta violar las reglas que establece el sistema de multitareas. Estas reglas son las siguientes:

- Todos aquellos procesos que deseen modificar los vectores de interrupción quedan excluidos de poderse ejecutar dentro del sistema de multitareas. Si se detecta la intención de algun cambio de vector (función 25H) el mismo sistema procede a matar al proceso mediante la llamada de la función (void congela (void)).
- Todos aquellos procesos que desean finalizar con la función (4cH) que indica fin de proceso con retorno de carro son congelados.
- Todos aquellos que utilizan la función (30H) para obtener la versión del DOS son congelados.

De las funciones implantadas en el lenguaje turbo "C" y de las más comunes tales como: printf(), scanf(), getchar(), cgets() y todas aquellas que crean, abren,cierran y borran archivos, tienen acceso a los servicios del sistema que proporciona el MS-DOS a través de las funciones de la interrupción 21H.

Ya que estas funciones llaman a sección crítica (código no reentrante), entonces hay que establecer un mecanismo para manipular las funciones de la interrupción 21H, para que sean útiles en un sistema reentrante.

La función cap_21 es diseñada con tal propósito. Monitorea cual de las funciones anteriores esta solicitando entrar al sistema y toma las medidas necesarias para manipularlas de manera adecuada.

Si un proceso hace uso de la función printf() de "C" y hace uso de cualesquier otra de las funciones que imprimen en pantalla que no sea la función 40H de la interrupción 21H del MS DOS, la función cap_21 transfiere el control a una sección de código (impresión) que realiza la impresión en pantalla en forma casi directa.

Luego, se pone el candado antes de entrar a la sección crítica, esto es antes de llamar a la interrupción 21H (int 81H). Se obtiene el estado del registro de banderas proporcionado por MS-DOS. Se verifica si algún proceso desea entrar a la sección crítica, si es así se conmuta, si no se quita candado y termina.

Cuando se ve involucrada la función 40H se tiene que verificar a que tipo de dispositivo se dirige la impresión. El dispositivo o periférico se señala mediante el valor de un apuntador "handle". Si el apuntador es menor que 05H, la impresión es dirigida hacia un dispositivo estandar de salida tal como la pantalla o la impresora y se realiza en forma directa como se acaba de describir. Si el apuntador es mayor o igual a 05H, el _cap21 transfiere el control a una sección de código reservada para manipular los problemas que se presentan con los periféricos.

Antes de entrar a sección crítica la cual estará involucrada con los problemas que se presentan con los periféricos, se llama a la interrupción creada "**int 82H**". Esta interrupción **esta encargada de sacar una "foto", esto es, obtener en pila una copia de todo el contexto y ambiente del estado de el proceso justo antes de entrar a la sección critíca. Carga los valores de los registros segmento de pila (SS) y apuntador de pila (SP), de la pila de trabajo actual en los respectivos campos reservados para estos valores (ss_peligro y sp_peligro) en la correspondiente tabla de control del proceso. Coloca una copia de los registros de banderas, registro de segmento de código (CS) y registro apuntador de programa (IP) en la cima de la pila.** Proporcionando así la posibilidad de reintentar la entrada nuevamente a la sección crítica cuando se registra un problema con periféricos.

El candado es puesto (candado = 0) y se invoca a la interrupción int 81H para entrar a la sección crítica. Si no se llegara a presentar algún problema (tiempo excedido) con el periférico dentro de sección crítica, se llama a la interrupción "**int 83H**" **quita foto. Esta interrupción realiza el trabajo inverso de la interrupción saca foto (int 82H), ignorando valores salvaguardados en pila que representaban el contexto y ambiente del proceso antes de entrar a sección crítica, posicionando los valores del registro de banderas, registro de código de segmento (CS) y registro apuntador de instrucciones (IP) del punto inmediato a seguir después de la llamada a la interrupción 21H (int 81H) del MS_DOS.**

Luego, se quita el candado (candado = 1) dejando que otro proceso entre a sección crítica.

Si se presentase un problema con algún periférico o tarda mucho (la correspondiente tajada de tiempo más un máximo de segundo y medio) en responder el MS-DOS al sistema de multitareas, el proceso es interrumpido por el módulo despachador (_asr) llamando a la interrupción **int 84H. Esta interrupción (int 84H) es la encargada de regresar al proceso al punto exacto antes de entrar a la sección crítica (entrada al MS-DOS al llamar a la int 81H). La cual obtiene los valores de los registros segmento de pila (SS) y registro apuntador de pila (SP) (ss_peligo,sp_peligro) desde la tabla de control del proceso que se interrumpio por exceso de tiempo.**

Luego, llama a una **función que despliega un mensaje de error (void recupera_recursos (void)) y dá un cierto tiempo para que el usuario corrija el error.** Restablece todos los valores de los registros del proceso interrumpido con la copia "fotografiada" por la interrupción "int 82H", antes de que se detectara un error por tiempo excedido en atención a un periférico, involucrado con sección crítica y se intenta de nuevo.

El procedimiento antes descrito, se aplica de igual manera en el caso en que el cap_21 detecte la intención de crear, abrir, cerrar o borrar un archivo dirigido generalmente a disco.

El tratamiento de cap_21 orientado a las entradas provenientes del dispositivo estandar (teclado) es el siguiente. El cap_21 contínuamente monitorea que tipo de servicio de entrada es requerido por el proceso. Esto es, verifica si se esta solicitando una entrada con la función cgets(), getchar() o scanf() de lenguaje "C". Por lo tanto se implantó una sección de código que realiza la rutina de checar el teclado.

Debido a que el lenguaje Turbo "C" tiene su propio y único espacio de memoria reservado para la captación de entradas por teclado, el cual la maneja en forma muy particular. Se procede a salvaguardar en pila la dirección de este espacio de memoria de teclado de "C". Entonces se procede a obtener desde la tabla de control respectiva, la dirección de memoria de teclado del proceso, para ir captando las entradas de teclado (unsigned char *espacio_teclado;).

Si el recurso del teclado aun no se le ha asignado al proceso que solicita entradas de datos por teclado (tarea_actual != tarea permitida) se conmuta al siguiente proceso y cede su tajada de tiempo. Eventualmente le toca de nuevo su turno, verifica nuevamente si ya se le asignó el teclado, si no, se repite el proceso anterior.

Si ya se le ha asignado el recurso del teclado (tarea_actual=tarea_permitida), entonces se transfiere el control a una sección de código dentro del cap_21 que realiza la atención al teclado.

Esta sección de código revisa si hay datos en el espacio de memoria del teclado a través de la función 44H de la interrupción 21H del MS-DOS, poniendo el candado antes de entrar a sección crítica. Al hacer la revisión verifica si algún otro proceso desea entrar a sección crítica (bandera = 1), si es así, conmuta de proceso quitando el candado (llamada a int 85H), si no, quita candado (candado = 1) y revisa si ya hay datos. Si aun no hay, entonces, conmuta a otro proceso (int 85H) cediendole su tajada de tiempo al siguiente proceso.

Si ya hay dato, entonces procede a tomar el dato a través de la función 01H de la interrupción 21H (int 81H) poniendo el candado justo antes de llamar a la int 81H. Verifica si algún proceso desea entrar a sección crítica si es así, conmuta a otro proceso y quita candado (candado =1). Esta parte de código de la función cap_21 revisa si el dato que se recibe es el carácter, espacio hacia atrás (código ASCII 08H), borrando los anteriores carácteres y descartándolos como carácteres inválidos.

Si es dato válido, entonces lo coloca en la dirección del espacio de memoria del teclado y espera por la llegada de retorno de carro (0dH) para incluir el carácter fin de linea (0aH). Se verifica nuevamente si el recurso de teclado esta asignado al proceso (tarea_actual = tarea_permitida.), si no esta asignado, entonces espera su turno y conmuta al siguiente proceso (int 85h), cediendo su tajada de tiempo.

Si se le asignó el teclado y (tarea_actual = tarea_permitida) es verdadero entonces, la información que se retuvo en la dirección del espacio de memoria del teclado particular de el

Por otra parte existen las rutinas de manipulación de carácteres como lo son entre otras: lectura y escritura de el carácter y sus atributos en la posición actual del cursor (función 08H y 09H).

El cap_10 se encarga de monitorear por cual de estas funciones, algun proceso, desea manejar el despliegue de su información en pantalla. El área de despliegue de información por cualesquiera de las funciones anteriores de la interrupción 10H del BIOS, esta regulada por los valores que se encuentran en la tabla de control de cada uno de los procesos; delimitando su despliegue de acuerdo a los valores que actualmente se encuentran en las variables que se definieron como límites de posicionamiento de cursor del proceso , (cursor_sup_limite_actual, cursor_inf_limite_actual, cursor_izq_limite_actual y cursor_der_limite_actual).

Es así como se manipula cada una de las funciones de la interrupción 10H del BIOS, delimitando su acción en forma independiente para cada proceso asignado a su propia ventana.

Si se registra la intención de realizar un desdoblamiento (función 06H) de la página activa hacia arriba, esto indica que el cursor se encuentra en el límite inferior de la ventana, por lo tanto la función cap_10 procede a hacer el desdoblamiento hacia arriba de el área correspondiente. Esta área (ventana) es acotada por los valores de los límites de la ventana así como sus atributos asignados al proceso, extraidos de su tabla de control correspondiente.

La mayoría de las funciones de manipulación de carácter, involucran el posicionamiento del cursor. De aquí que, la función cap_10 continuamente monitorea la manifestación de la función 02H del BIOS.

Si se registra alguna petición de posicionamiento de cursor (función 02H), se realizan comparaciones continuas entre el punto en que se quiere colocar el cursor y los valores que delimitan su ventana.

Si se llega a comprobar que el punto en que se desea colocar el cursor es menor que el valor del límite izquierdo (columna izquierda) asignado a la ventana (cursor_izq_limite_actual), se procede a decrementar el renglón y se ajusta a límite izquierdo (columna izqierda). Se actualizan los valores de la posición del renglón dentro de la tabla de control correspondiente y se procede con su impresión en pantalla.

Si el punto donde se quiere colocar el cursor es mayor o igual que el límite izquierdo (columna izquierda), se procede a comparar con el límite derecho (columna derecha). Si se comprueba que el punto donde se desea posicionar el cursor es menor o igual que el límite derecho, entonces implica que se encuentra dentro de su ventana. Se monitorea si se ha recibido un retorno de carro, si no, se procede a actualizar los valores de la posición actual del cursor en la tabla de control correspondiente y se procede con la impresión en pantalla. Pero si se registró retorno de carro, entonces se actualiza el posicionamiento izquierdo del cursor, de acuerdo a el valor del límite izquierdo, extraido del parámetro (limite_izquierdo) de su correspondiente tabla de control y se procede con la impresión en pantalla.

Si se comprueba que el punto donde se desea posicionar el cursor es igual o mayor que el límite derecho de su ventana, se actualiza el posicionamiento del cursor con el correspondiente valor del límite izquierdo de su ventana.

Se monitorea si ya se ha llegado al fondo de la pantalla, esto es se compara el posicionamiento vertical del cursor (cursor_inf_limite_actual). Si no se ha llegado al límite inferior de la ventana, entonces se incrementa el renglón, se actualizan los valores de la posición actual del cursor en la tabla de control y se procede con la impresión. Pero si se registra que se ha llegado al fondo de la pantalla. Entonces se procede a tomar los atributos de la ventana y se hace un desenrollamiento de la pantalla hacia arriba. No se incrementa el renglón y se procede con la impresión en pantalla.

Los límites de las ventanas se dejan como responsabilidad del usuario. se sugiere no sobreponer ventanas para evitar la mezcla de la información desplegada en pantalla.

V.4.- Procesador de mandos.

Debido a que el teclado esta estrechamente relacionado con el usuario, se intercepto la interrupción 09H del teclado.

El teclado mediante una serie de mandos (teclas o combinaciones especiales de teclas) le proporciona al usuario la capacidad de interactuar con el sistema de multitareas con ambiente de ventanas.

El vector de la interrupción 09H del teclado se interceptó y se dirigió hacia la función cap_09. Esta es la función encargada de manipular el teclado, analizando los códigos de exploración.

El código de exploración es un número de un byte de largo, que en la IBM-PC y (XT) tiene la siguiente interpretación: Cada una de las teclas genera un número al ser presionada y otro al ser soltada. Los siete bits más bajos representan un número de identificación arbitrario asignado a cada tecla. El octavo bit igual a "1" indica que ha sido presionada la tecla y si es igual a "0" ha sido soltada. En la IBM (AT) el código de exploración es el mismo al presionar la tecla que al soltarla, solo que en el caso de soltarla le precede un segundo byte con un valor de "F0H".

Los códigos de exploración en la IBM-PC y (XT) son colocados por el microprocesador del teclado (8048), en el puerto A (60H) del circuito integrado controlador de periféricos (8255). La IBM (AT) no utiliza el circuito integrado 8255, sin embargo utiliza la misma dirección de puerto (60H) para poner o consultar la información de los periféricos instalados.

La función cap_09 monitoréa el estado del teclado, teclas y combinación de teclas que pueden producir efectos indeseables en un sistema de multitareas.

De las teclas que se eliminaron, e independientemente que estén en mayúsculas o minúsculas están las que se combinan con la tecla de Ctrl que son: S, P, 2, C, G. La tecla del Num Lock, la de "PAUSE" en teclado extendido. Las combinaciones de Ctrl con Print Screen en teclado extendido. Y las teclas en combinación con la tecla Alt que son: Q, R.

Otras fueron interceptadas (F1, F2, F3, F4, F5, F6) con la finalidad de formar macros que ejecuten funciones bien definidas dirigidas al procesador de mandos (tarea número cero) del sistema de multitareas. Otras más fueron interceptadas con la finalidad de modificar los límites izquierdo y derecho de las ventanas, proporcionándole al usuario la capacidad de aumentar o disminuir horizontalmente el área de despliegue de la ventana.

Si la tecla que elige el usuario es alguna de las que se redirigieron hacia el procesador de mandos (de la F1 a la F6) cada una de estas, actualiza el valor de una variable (pregunta), indicando al procesador de mandos que realice cierta acción específica.

Si es presionada la tecla F1, el procesador de mandos proporcionará al usuario la capacidad de hacer visible el proceso que él elija. Si es presionada F2, el usuario tiene la capacidad de ocultar el proceso. Si es presionada F3, el usuario puede desactivar a un proceso. Si es presionada F4, el usuario puede reactivar un proceso. Si es F5, el procesador de mandos le da la capacidad de conmutar a la tarea elegida (esto es, interactuar con el proceso mediante el teclado).

Cuando es elegido interactuar con un proceso mediante el teclado y al finalizar con él, se puede regresar al procesador de mandos presionando conjuntamente las teclas Ctrl-Break.

Si es F6, el procesador de mandos proporciona la función que da la posibilidad de salir del sistema de multitareas. Al presionarse ésta tecla el procesador de mandos despliega un mensaje, preguntando si se desea terminar con el sistema de multitareas. Si es afirmativo se invoca a una función (_termina_multitareas) que se encargada de redirigir a su dirección original, a todos los vectores de interrupción que fueron modificados al inicio del sistema de multitareas. Se deshabilita al UART y se regresa a la pila original de turbo C , para que sea él mismo turbo C el encargado de regresar todas las partes que el modificó. De esta manera se restablece el ambiente normal del sistema.

Para modificar los límites de las ventanas es necesario que el teclado este asignado a la ventana que está interactuando con el usuario.

Para aumentar el área de la ventana hacia la derecha se obtiene con la combinación de las teclas Alt (->), llegando como máximo al extremo derecho de la pantalla.

Para disminuir el área de la ventana la izquierda, se usa la combinación de las teclas Ctrl y (<-). Llegando como máximo al límite izquierdo establecido menos uno (cursor_izq_limite - 1).

Para aumentar el área de la ventana hacia la izquierda, se controla con la combinación de las teclas Alt y (<-).

Para disminuir el área de la ventana hacia la derecha, se controla con la combinación de las teclas Ctrl y (->).

El estado del teclado se registra en la palabra de control del teclado que se encuentra en la localidad del BIOS 0040:0017 para la IBM PC y XT. Esta palabra de control continuamente es monitoreada por la función cap_09, para revisar bajo qué condiciones propuestas por el usuario se encuentra el teclado. Esto es, si las teclas Ins, Caps Lock, Num Lock, Scroll Lock, se encuentran activadas o si las teclas Alt, Shift izquierdo o derecho fueron momentaneamente presionadas, al efectuarse una entrada de datos al sistema.

Si la combinación de teclas editadas por el usuario corresponde a las elegidas para realizar las modificaciones de los límites de las ventanas (Alt y (->), Alt y (<-), Ctrl y (<-), Ctrl y (->)), la función cap_09 lo registra y redirige la ejecución del apuntador de programa con una llamada lejana a una función en lenguaje "C" (limites_der_ventanas, limites_izq_ventanas, limites_der_ventanas_menos, limites_izq_ventanas_menos) para que realice las modificaciones a los límites, ponga los atributos de las ventanas y actualice estos valores en la correspondiente tabla de control.

Todas estas teclas que se mencionaron pasan a través de la función cap_09. Si son útiles, se hace uso de ellas y se eliminan, si no, solamente se eliminan, enviando un mando de reconocimiento al microprocesador del teclado (8048) para que se active de nuevo y un mando al controlador de interrupciones (8259) para que acepte interrupciones.

Todas aquellas teclas que no intervienen o interfieren con el sistema de multitareas son pasadas sin ninguna modificación.

VI.- CONCLUSIONES Y RECOMENDACIONES.

Las amplitudes y extensiones en el desarrollo de programas pueden ser tantas como la imaginación y el tiempo lo requiera. El diseño e implantación del sistema de multitareas en tiempo real con un ambiente de ventanas en modo texto, para las computadoras personales

IBM, no es la excepción. Se pueden desarrollar sobre él muchas más habilidades y con gran flexibilidad. Sin embargo el sistema de multitareas se diseñó y se implantó hasta cierto límite, en el que se reunieron los requerimientos del objetivo principal del trabajo.

Como resultado se obtuvo:

-- Un sistema de multitareas para utilizarse en computadoras personales IBM o 100 % compatibles que corre sobre la cima del sistema operativo MS-DOS.

-- Se desarrolló un mecanismo que permite a los programas de aplicación ejecutarse en tiempo real interviniendo la interrupción 08H.

-- Se desarrolló una interfaz con el usuario que proporciona una manera sencilla y rápida de interactuar con el sistema de multitareas.

-- Un manejador de video que proporciona la habilidad de trabajar con un ambiente de ventanas en un sistema de video CGA.

-- Un manejador de teclado que provee el mecanismo para interactuar con el usuario via la entrada estandar de la computadora mediante un procesador de mandos.

-- Se desarrollo un mecanismo que permite que el sistema operativo MS-DOS pueda ser empleado en un sistema reentrante, mediante un manejador de las funciones principales de la interrupción 21H.

-- El acceso al sistema mediante los servicios que proporcionan las llamadas al sistema operativo tanto del BIOS como del MS-DOS, la intervención y manipulación de las rutinas de atención a las interrupciones, la creación de rutinas propias de atención a interrupciones, así como el uso de puertos de comunicación, fueron los mecanismos de trabajo para el desarrollo del sistema de multitareas.

Restricciones:

-- Se restringe el uso de ciertas funciones del lenguage C por ser no reentrantes. Tenemos así, que las funciones: malloc(), free(), fopen(), fclose(), deben ser actualmente reemplazadas por malloc_new(), free_new(), fopen_new() y fclose_new().

-- Otras funciones del lenguage C pueden presentar el problema de no reentrancia, como las que se acaban de mencionar, sin embargo el tratamiento de como evitar esta situación se puede resolver con el mismo método que se siguió en el programa para las funciones de malloc(), free(). Esto es utilizando un semáforo, para que ningun otro proceso utilize el mismo código al mismo tiempo, etc.

-- Los límites de las ventanas se dejan como responsabilidad del usuario, sugeriendo no sobreponer ventanas para evitar la mezcla de la información desplegada en pantalla.

-- Los programas que se desén ejecutar en el sistema de multitareas deben ser compilados junto con el programa principal.

-- La ejecución del sistema de multitareas sobre equipos de computo que no sean 100% compatibles con IBM puede producir un mal funcionamiento del mismo.

Los principales problemas que se presentaron durante el desarrollo del sistema de multitareas fueron entre otros:

-- La falta de información más completa correspondiente al sistema operativo MS-DOS.

-- La no reentrancia del MS-DOS.

-- Falta de información técnica del lenguaje turbo C.

-- La no reentrancia de muchas funciones del lenguaje turbo C.

-- La falta de estandarización entre los diferentes dispositivos involucrados en sistemas que aparentemente son 100% compatibles con IBM.

-- La falta de servicios del sistema destinados a manejo de periféricos.

-- En algunos sistemas de arreglo de video a color (VGA) no respeta los límites establecidos para las ventanas.

Como recomendaciones:

Entre otras existe la posibilidad de continuar con este trabajo incluyendo:

-- El cargado de programas, DOS virtual, trueque a disco, desarrollo de ventanas tipo "X Windows", comunicación entre procesos más completo.

-- Al realizar el cargado de programas, se podria instalar un DOS independiente a cada uno de los procesos eliminándose el problema de no reentrancia del MS-DOS. Se utilizaría el procesador de mando del mismo MS-DOS para ejecutar desde el interior del sistema de multitareas las funciones que provee DOS para el manejo de los archivos, como seria el copiado, el "formateo", el respaldo de disco etc.

Sin embargo este esquema plantea nuevos problemas que resolver, como lo sería un espacio de memoria más grande para cada uno de los procesos y si llegan a estar demasiados procesos en memoria principal, puede exceder el límite de la memoria convencional. Por lo tanto hay que desarrollar un manejador de memoria que pueda accesar todas aquellas localidades de memoria que se encuentran por encima del 1Mb de memoria.

Además se requiere la creación de sistemas de arbitraje o señalización que permitan manejar los periféricos de una manera efectiva. Para que no se presenten los problemas entre otros los de abrazo mortal, de carreras y secciones críticas.

-- El trueque de programas inactivos de memoria principal a disco y viceversa.

-- Interfaz usuario por medio de icons.

-- La posibilidad de manejar periféricos mediante el desarrollo de periféricos virtuales.

-- Extender la habilidad del sistema de multitareas programando a los microprocesadores 286 y 386 a que trabajen en modo protegido, virtual y por paginación respectivamente, obteniendo todas las ventajas que nos proporcionan estas pastillas si las trabajamos al máximo.

APENDICE.

Interrupción 10H del BIOS: Servicios de video.

Para poder controlar y manejar el ambiente de video se interceptó la interrupción 10h ; servicios básicos de video, que proporciona el BIOS.

En particular para el diseño del ambiente de ventanas se necesita saber como tratar y manejar el despliegue de video así cómo, obtener acceso a otros servicios del sistema operativo.

Las funciones de la interrupción 10h del BIOS se describen en IBM technical reference 1986.

Se utilizó el modelo de ventanas múltiples orientado a texto para manejar el visualizador de pantalla. Este modelo organizá a la pantalla en áreas separadas (ventanas) y cada programa tiene su correspondiente ventana.

Todos los sistemas de video utilizan espacio de memoria reservada desde el cual la imagen de la información que está siendo puesta en dicha memoria, es deplegada en pantalla, que es periódicamente actualizada tomando los datos del área de memoria reservada a video.

La localidad de memoria designada a video varía de sistema a sistema y es de acuerdo al modo de operación de la pantalla y a la cantidad de memoria designada o dedicada a éste propósito.

Cuando se mantienen imágenes múltiples de pantalla en el área de memoria de video, cada imagen constituye una página.

Una página puede ser de 4KB de memoria para una pantalla de 25 renglones por 80 columnas. Por ejemplo, una IBM con un adaptador monocromático tiene 4k bytes de memoria reservada, iniciando en la dirección absoluta B0000H. Este es el espacio de memoria suficiente para desplegar una pantalla de 25 renglones por 80 columnas de texto.

El adaptador de gráficas a color CGA tiene 16K bytes de memoria reservada en la memoria RAM, iniciando en la dirección absoluta de la memoria convencional B8000H. Esta es la memoria suficiente para una pantalla en modo gráfico, sin permitir paginación o de cuatro a ocho pantallas en modo texto, dependiendo si es de 25x80 ó 25x40 renglones/columnas.

El adaptador de gráficos extendido EGA puede estar equipado con 64K, 128K ó 256K bytes de RAM. La dirección de inicio es programable. Puede iniciar en la localidad A0000H para modos de gráficos avanzados y en B0000H y B8000H por compatibilidad con los modos monocromáticos y CGA. El número de páginas disponibles depende tanto del modo de operación de la pantalla y en base a la cantidad de memoria presente.

En modos de texto el área de memoria inicia desde el renglón superior de la pantalla, de izquierda a derecha, la continuación de los datos sigue desde el fin derecho de un renglón al inicio izquierdo del renglón siguiente, tal como si toda la pantalla fuese un solo renglón.

Todos los sistemas de video operan en la misma forma cuando estan en modo texto. Son asignados 4000 bytes de tal forma que hay dos bytes para cada una de las 2000 posiciones de la pantalla (25 renglones X 80 columnas). El primer byte tiene el código ASCII,el segundo byte contiene la información de como debe aparecer el carácter, es decir el byte del atributo.

La circuitería de video convierte el número ASCII a su símbolo asociado, así como su atributo y lo envia a pantalla.

Interrupción 09H: Manejo del teclado.

El teclado estandar de la IBM contiene un microprocesador dedicado, 8048 INTEL, el cual detecta cualquier acción sobre el teclado y reporta cualquier actividad de éste a la UPC a través del controlador de interrupciones programable 8259-A.

Cuando una tecla es presionada, el 8048 deposita un código de exploración en el puerto "A" del circuito de interfaz de periféricos Intel 8255 , localizado en la tarjeta del sistema y genera una interrupción por circuitería llamada IRQ1, la cual tiene el segundo nivel de prioridad más alto de todas las interrupciones por circuitería de la PC.

El 8259-A le indica a la UPC que una señal por la línea de interrupciones IRQ1 se ha recibido. La UPC transfiere el control a la interrupción 09H, la rutina del BIOS de atención al teclado y se lee el código de exploración. Esta rutina es la responsable de convertir el código de exploración a código ASCII, leer la información del registro de estado del teclado, la ejecución de la pausa (Pause o Ctrl-NumLock), interrupción 05H (Print screen) y reinicializado (Ctrl-Alt-Del).

Después de decodificar el teclazo, la interrupción 09H almacena tanto el código de exploración y el código ASCII en un área de memoria circular localizado en el área de datos del BIOS. Este espacio es de 16 palabras (32 bytes) de largo y puede retener 16 teclazos (ya que cada teclazo necesita dos bytes).

Las entradas son finalmente leidas cuando el programa llama a la interrupción 16H del BIOS.

Un código de exploración es un número de un byte de largo, en el cual los siete bits menos significativos representan un número de identificación arbitrario asignado a cada tecla. Exceptuando la IBM- PC AT, el bit más significativo (bit 7) indica cuándo la tecla ha sido presionada (bit 7 = 1,"make code") o cuando ha sido soltada (bit 7 = 0, "break code"). Ejemplificando, el código de exploración de la tecla "B" es 48, el cual es en binario 110000.

Cuando la tecla va hacia abajo, el código enviado al puerto "A" de 8255 es 10110000 y cuando va hacia arriba es 00110000. Así cada teclazo se registra dos veces en el puerto "A" del 8255. En la IBM-PC AT trabaja ligeramente diferente, enviándose el mismo código en cualquier caso, pero le precede un byte F0h cuando la tecla es soltada.

Cada vez que un código de exploración es colocado en el puerto "A", el 8255 envia una señal de reconocimiento al microprocesador del teclado 8048 y una interrupción 09H es llamada. La UPC deja de hacer momentaneamente su trabajo y realiza una rutina de atención a la INT

09H, que analiza el código de exploración. Un cambio en el estado de las teclas es registrado en memoria, cuando este código se origina en combinación de las teclas Shift, Ctrl, Alt, flecha izquierda y flecha derecha o CapsLock, NumLock y Scroll Lock.

La dirección del registro del estado del teclado se encuentra en la dirección 0040:0017H (para la IBM PC y XT), que es donde se registra si un tipo de éstas teclas se encuentra presionada constantemente ("ON") , soltada ("OFF") ó temporalmente presionada.

La rutina de atención a ésta interrupción revisa primero el registro de estado del teclado para obtener el código de carácter ASCII ("American standard code for information interchange") correcto (por ejemplo para ver si es minúscula o mayúscula). Entonces el código ASCII es colocado en el espacio de memoria reservado al teclado, el cual contiene un área de memoria que almacena hasta 15 carácteres.

Hay dos tipos de códigos de carácteres utilizados por la IBM-PC; ASCII extendidos para la IBM y códigos extendidos.

Los códigos ASCII extendidos para la IBM son números de un byte que corresponden al conjunto de carácteres que incluyen los símbolos usuales de una máquina de escribir americana más un grupo de teclas especiales y símbolos gráficos. Los ASCII incluyen también códigos de control ordinariamente utilizados para enviar mandos a periféricos que son diferentes a los que actúan como carácteres en la pantalla. En concreto, solo los primeros 128 carácteres son verdaderos carácteres ASCII.

El segundo tipo de código, el código extendido, se asigna a teclas o combinaciones de teclas que no tienen símbolo definido en el código ASCII que los represente, tales como las teclas de funciones y combinación de teclas con "Alt".

Los códigos extendidos son de dos bytes de largo y el primer byte es siempre un código ASCII. El segundo byte es el número de código.

Existen algunas combinaciones de teclas que realizan funciones especiales y que no generan código de exploración. Estas combinaciones incluyen <Ctrl-Break>, <Ctrl-Alt-Del> y <Prt Scr>, además <Sys req> en la AT. Estas combinaciones representan acciones especiales

predefinidas. Además existen combinaciones de teclas alfanuméricas y con "Ctrl" importantes para DOS, como son <Ctrl-C>, <Ctrl-G>, <Ctrl-H>, <Ctrl-M>, <Ctrl- P>, <Ctrl-S> , <Ctrl-Z>, etc.

Interrupción 14H: Servicios del puerto serie.

La IBM-PC utiliza como elemento de comunicación asíncrona el trasmisor receptor asíncrono universal ("UART" por sus siglas en inglés) INTEL 8250. Este circuito utiliza los servicios de interrupción del controlador de interrupciones programable (PIC) INTEL 8259-A.

Toda la comunicación entre el UART y UPC es realizada a través de un grupo de puertos de entrada y salida reservados para éste propósito. El MS-DOS, hasta la versión 3.30 maneja dos puertos de comunicación y de aqui que puede controlar la comunicación de dos UART.

Las direcciones de base estan en 0040:0000H para la comunicación del primer UART (COM1) y en la 0040:0002H para el segundo UART (COM2). (Una dirección de base es la dirección más baja de un grupo de direcciones de puertos mediante los cuales el UART puede ser accesado). En casi todas las máquinas el COM1 inicia en la dirección de puerto 3F8H y el COM2 en la 2F8H.

Un UART 8250, tiene diez registros programables de un byte de largo, mediante los cuales se controla y se monitorea el puerto serie. El acceso a estos diez registros es a través de siete direcciones de puerto que son de la 3F8H - 3FEH (ó 2F8H - 2FEH).

Cinco de los diez registros, tienen su acceso dependiendo del séptimo bit del registro de control de líneas, el cual es el único registro que se encuentra en la dirección de puerto 3FBH. De los diez registros, solo seis son nesesarios para la comunicación básica en serie y los cuatro restantes incluye dos para el control del modulador-demulador de señales "Modem" y dos registros relacionados con comunicación por interrupción.

Debido a que la UPC opera mucho más rapido que los 300 a 1200 bits por segundo con que opera tipicamente la comunicación en serie, el método de comunicación por interrupciones es más eficiente, evitando el desperdicio de tiempo de la UPC que podría representar al estar monitoreando constantemente al puerto serie. Por ésta razón es conveniente que el UART 8250 se configure para obtener los datos via interrupción, siempre que llegue un carácter, un error, etc.

La interrupción pone en acción un procedimiento para que realice la manipulación de entrada-salida de los datos en forma conveniente.

Durante la inicialización de un puerto de comunicación se establecen todos los parámetros mediante los cuales opera. Estos parámetros incluyen el bit de inicio, la longitud de la palabra, el número de bits de parada, la configuración de paridad y la velocidad de transmisión (baud rate).

La interrupción 14H del BIOS proporciona los servicios para la comunicación via puerto serie. La función 00H de la interrupción 14H del BIOS, inicializa el puerto serie. En DX se le proporciona el número del canal de comunicación (COM1=0, COM2=1), en AL se le da el byte que da la información de inicialización, (IBM technical reference, 1986).

Interrupción 05H: Vaciado de la pantalla a la impresora.

En un teclado de 83 teclas cuando la tecla "PrtScr" es presionada produce un asterisco (ASCII 42). Cuando es presionada en conjunto con la tecla <Ctrl> envia un código extendido 114, pero cuando es presionada en combinación de <Shift> + <PrtScr> representa en si misma un resultado especial (en el teclado de 101-102 teclas con el solo hecho de presionar la tecla <Print Screen>): imprimir el contenido de la pantalla en la impresora.

Cualesquier otro teclazo provoca que la interrupción del teclado (INT 09H) deposite los códigos de exploración en el espacio de memoria reservado para el teclado y registra el estado para las teclas llamadas "toggles" las que permanecen siempre en estado de encendido o apagado durante algun tiempo y "Shift" que temporalmente son presionadas.

Un teclazo puede no tener impacto alguno hasta que el programa en proceso decida leerlo desde espacio de memoria reservado para el teclado. Pero la combinación <Shift> + <PrtScr> (<Print Screen> para el teclado de 101-102 teclas) provoca que la interrupción del teclado transfiera el control a la rutina que esta siendo apuntada por la interrupción 05H (INT 05H). En este sentido funciona como una interrupción por circuitería.

Interrupción 08H: Temporizador o Cronometro (Timer).

El temporizador 8253 INTEL tiene tres canales independientes idénticos y cada uno puede ser programado de manera conveniente. El 8253 opera independientemente de la UPC, y solo emplea la UPC para programar el circuito, por lo tanto funciona como un reloj en tiempo real, manteniendo su ritmo independientemente de lo que suceda con la computadora mientras este funcionando.

La razón de entradas del reloj del sistema hacia el temporizador es de 1,193180 veces por segundo y tomando en cuenta que el número más grande que puede contener un registro de 16 bits es 65535 valor el cual puede ser empleado como divisor de la razón de entradas. Por lo tanto puede producir salidas de pulsos de reloj a una razón de 18.2 veces por segundo, obteniéndose como periodo más largo posible entre pulsos de aproximadamente 1/20 vo. de segundo.

Por el canal 0 del temporizador 18.2 veces por segundo invoca una interrupción por circuitería la INT 08H, la cual detiene brevemente a la UPC e incrementa una variable que se encuentra en el área de datos del BIOS llamada, contador del tiempo del día, esta cuenta se registra en la localidad de memoria 0040:006CH.

El canal 1 controla el refrescamiento de memoria y nunca debe ser interferido.

El canal 2 esta conectado a la bocina de la computadora y éste produce simples señales para hacer sonido.

El BIOS contiene una interrupción "boba" especial, la INT 1CH, la cual no hace nada hasta que se le proporciona una rutina. Al inicio, el vector apunta a una instrucción IRET. Lo que hace especial a esta interrupción es que es invocada por la rutina del temporizador (timer) del BIOS, después que ha actualizado el contador del tiempo del día. Esto es una interrupción por circuiteria que ocurre 18.2 veces por segundo. Se puede cambiar tal que este vector apunte a un procedimiento y entonces tal procedimiento será llamado 18.2 veces por segundo (digamos a un procedimiento de asignación de recursos).

Interrupción 1BH: Suspensión de programa.

Cuando se presiona la combinación de las teclas <Ctrl> + <Break>, la interrupción del teclado pone una bandera, indicando que la rutina de atención al <Ctrl-Break> INT 1BH entre en acción. El control se le otorga a la rutina de atención solamente cuando el programa utiliza una función del DOS que es capaz de censar esta bandera. Normalmente, sólo las funciones de entrada y salida del DOS pueden detectar la bandera del <Ctrl-Break> (las funciones 01H - 0CH de la INT 21H excluyendo a las 06H y b07H).

La rutina del <Ctrl-Break> existe como una forma de salirse del programa actual en cualesquier momento. Cuando una función del DOS detecta el estado de la bandera del <Ctrl-Break>, el control es dirigido a la rutina apuntada por el vector de interrupción 23H entonces DOS ejecuta una rutina de finalización de programa.

LITERATURA CITADA.

Biggerstaff, T. J. [1985]. System software tools. Prentice-Hall, Englewood Cliffs, New Jersey.

Brawer, S., [1989]. Introduction to parallel programming. Academic Press. Boston, Massachusetts.

Deitel, H. M., [1988]. An introduction to operating systems. Addison-Wesley. Massachusetts.

Dijkstra, E. W., [1965a]. Cooperating secuential processes. Technical report EWD-123, Technological University, Eindhoven, The Netherlands, (1965); reprinted in [Genuys 1986], 43-112 pp.

IBM Technical reference [1986]. Personal computer Hardware reference library. International Business Machines Corporation Press, Florida.

Jourdain, R., [1986]. Programer's problem solver for the IBM PC, XT ,& AT. Brady books. Prentice-Hall Press. New York, NY.

Peterson, J. L., y A. Silberschatz. [1985]. Operating system concepts. Addison-Wesley Publishing Company, Inc. Massachusetts.

Ripps, D., [1990]. The nature of real time. EDN's software engineering special issue. September 17, 1990.

Scalon, L. J., [1985]. IBM PC & XT Assembly language. A guide for programers. Brady books. Prentice-Hall, Inc. N.Y.

Shaw, A., [1988]. Special section on operating system. March, 1988. Vol 31. No.3. Communication of the ACM.

Tanenbaum, A. S. [1987]. Operating systems: Design and implementation. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

The MS-DOS Encyclopedia [1988]. Microsoft Press, Redmond, Washington.

Tsichritzis, D. C., y P. A. Bernstein. [1974]. Operating systems. Academic press, New York.

Zavitzky, S. R., [1985]. Real-time microprocessor systems. Van Nostrand reinhold company inc., Ney York.