Modo Protegido del 80386

Es necesario explicar cada uno de los tres modos de funcionamiento que ofrece el 80386 para poder entender posteriormente cada uno de los pasos realizados para implementar a DMT. A lo largo de los tres capítulos siguientes se explicará cada uno de estos modos. No obstante, no se comentarán aquellos detalles que no se han utilizado de cualquiera de los tres modos anteriores para la elaboración de DMT. En [INTEL386] se explica más detalladamente los modos de trabajo del 80386.

- **Arquitectura del 80386**

Antes de entrar en detalle sobre el Modo Protegido vamos a comentar cuales son los diversos registros que contiene el procesador para poder realizar todas sus funciones.

Podemos clasificar los registros del 80386 en diversas categorías:

- Registros generales: Son registros de 32 bits utilizados fundamentalmente en operaciones aritméticas y lógicas.
- Registros de segmento: Estos registros contienen la dirección de un segmento de memoria, a partir del cual podemos referenciar cualquier dato que se encuentre en cualquier dirección de memoria.
- Registros de estado y de instrucción: Se usan para almacenar el estado del procesador tras la ejecución de una instrucción y para llevar un control sobre la instrucción que se va a ejecutar.
- Registros del sistema: Se suelen dividir en registros de estado extendido, registros de gestión de memoria, registros de control, registros de depurado y registros de test.
- **Registros de propósito general**

Los registros generales del 80386 son los registros EAX, EBX, ECX, EDX, EBP, ESP, ESI y EDI de 32 bits cada uno. Estos registros suelen ser utilizados para operaciones aritméticas, lógicas y cálculo de direcciones. Los resultados de diversas operaciones aritméticas afectan sólo a diversos registros, por lo que su contenido dependen por tanto del tipo de instrucción que se ejecute.

Cada uno de los registros anteriores se pueden dividir en uno de 16 bits y son referenciados con el mismo nombre pero sin la E, por ejemplo AX, BX, CX, etc. A su vez los registros de 16 bits, como AX, BX, CX y DX, se puede dividir en dos de 8 bits, referenciados por la parte alta y baja, quedando como AH, AL, BH, BL, CH, CL, DH, DL. Para realizar cualquier operación con ellos los podemos utilizar indistintamente como registros de 8, 16 ó 32 bits.

En la figura 4.1 se puede observar el conjunto de registros de propósito general y sus posibles divisiones.

Figura 4.1. Registros de propósito general

- **Registros de segmento**

Un programa suele estar dividido en diversos módulos (segmentos) que suelen contener los datos, código y pila. Para el acceso a cada uno de estos módulos necesitamos un registro de segmento que los referencie y así poder acceder a cualquier dato que contenga cada uno de estos módulos.

Los diversos registros de segmento son: CS, DS, ES, SS, FS y GS de 16 bits. Cada uno de estos registros especifican un tipo de segmento del programa. Así, por ejemplo, el registro CS apunta siempre al segmento de código actual, el registro SS apunta al segmento de pila, el DS a los datos y el resto suelen ser auxiliares para

realizar diversas operaciones de datos entre segmentos, o acceder a los datos al igual que DS.

- **Registros de estado e instrucción.**

El registro de estado, llamado EFLAGS, mantiene información sobre la ejecución de la última instrucción. Este registro de 32 bits de longitud indica en cada uno de estos bits un tipo de información. Por ejemplo, uno de estos bits (ZF) indica si el resultado de una operación aritmética ha dado como resultado cero. La información que contiene el registro EFLAGS suele ser empleada para realizar saltos condicionales a otros puntos del programa. En la figura 4.2 puede observarse la información contenida en EFLAGS.

El registro de instrucción (EIP) mantiene la dirección respecto al segmento de código (CS), de la instrucción que va a ser ejecutada por el procesador.

Figura 4.2. Registro EFLAGS

- **Registros de sistema**
- **Registros de gestión de memoria**: Estos registros mantiene información sobre ciertas estructuras de memoria que son utilizadas en el Modo Protegido. Los diversos registros de memoria son: GDTR, LDTR, IDTR, TR y son usados para fines distintos. El propósito de estos registros se comentará posteriormente.
- **Registros de control:** Estos registros llamados CR0, CR2 y CR3 tienen fines muy distintos. El registro CR0 contiene información sobre el modo de funcionamiento del procesador, por ejemplo, posee un bit que indica si el procesador está en modo protegido, otro indica si la paginación está habilitada, etc. El registro CR2 es usado en el caso de errores de paginación y CR3 indica la dirección donde se encuentra el directorio de páginas para realizar la traducción de direcciones lógicas a físicas.
- **Registros de depurado:** Utilizados para facilitar funciones a los depuradores.
- **Gestión de Memoria**

Bajo el Modo Protegido, cuando el programador referencia una dirección de memoria, ésta no se corresponde con la dirección física final de memoria. El procesador ofrece diversos mecanismos para que esto sea así y se pueda controlar en todo momento la ejecución de un programa en memoria sin que sea un peligro para el sistema.

El proceso que se sigue desde que el programador referencia una posición de memoria hasta que se accede a la dirección física de memoria es el siguiente. En la figura 4.3 se puede observar gráficamente este mecanismo. La dirección que indica un programador en su aplicación, por ejemplo 1234:0088, se llama **dirección lógica**. Esta dirección lógica se somete a una traslación de segmento y se convierte en una **dirección lineal**. Por último esta dirección lineal se convertirá en una **dirección física** tras la traslación de página.

- **Traducción de una dirección lógica a lineal**

Una dirección lógica tiene el formato *selector:desplazamiento*, por ejemplo 100:2322. Esta dirección, dada por el programador, se convierte en una dirección lineal a través del siguiente mecanismo, la traslación de segmento.

El *selector* de la dirección lógica sirve como índice en una tabla de descriptores (llamada GDT), el selector 1 correspondería con el descriptor 1 de la tabla de descriptores, el selector 2 correspondería con el descriptor 2 y así sucesivamente. Cada uno de estos descriptores aportará los datos suficientes al procesador para convertir una dirección lógica en lineal. Algunos campos del descriptor son:

- **Base:** Contiene la dirección de un segmento de memoria dentro de todo el espacio direccionable por el procesador, 4 gigabytes.

- **Límite:** Contiene la longitud del segmento.
- **Bit de granularidad:** Especifica las unidades de medida que se debe de interpretar el campo límite. Si el segmento es granular indica que la longitud del segmento es igual a Límite * 4096.
- **Bit de Presencia:** Indica si el segmento se encuentra en memoria actualmente.
- **Bit de acceso:** Indica si se ha accedido al segmento. Utilizado para memoria virtual.
- **Privilegios del segmento:** Indica el nivel de privilegio del segmento.

Con la información que hay contenida en el descriptor el procesador sabrá donde encontrar el segmento en memoria, sabrá si un usuario tiene permisos de acceso a él, etc. La dirección lineal estará formada por la concatenación del campo Base del descriptor + el desplazamiento dado en la dirección lógica. En la figura 4.4 se puede observar como funciona este mecanismo de traducción.

El registro GDTR del procesador contiene la dirección en donde reside la tabla de descriptores ó GDT.

figura 4.4. Traslación de segmentos

- **Traducción de dirección lineal a física**

Esta fase de transformación de direcciones implementa las características básicas para los sistemas operativos de memoria virtual orientados a páginas y protección a nivel de página.

La traducción de lineal a física es opcional. Así, si no tenemos activada la paginación, la dirección lineal obtenida en el mecanismo anterior de traducción será la dirección física final. Para saber si la paginación está o no activada debemos de mirar el bit PG del registro CR0.

Este mecanismo de traducción es ligeramente más complicado que el anterior, ya que tendremos dos niveles de direccionamiento, pero en cierto modo es bastante similar. Al igual que en la traducción de lógica a lineal necesitábamos una tabla de descriptores para obtener la dirección lineal, ahora necesitamos dos nuevas tablas, directorio de páginas y tablas de páginas, para lograr la dirección física final.

Para explicar este proceso de traducción debemos de definir el formato de una dirección lineal. En la figura 4.5 podemos ver este formato gráficamente.

Los 10 bits más significativos de la dirección lineal representan el número de entrada en el directorio de páginas. Este directorio está formado por diversas entradas que contienen información sobre la tabla de páginas a la que referencia (una tabla de páginas tiene un tamaño de 4Kb). Algunos campos clave del directorio son:

- **Dirección de la tabla de páginas**: Indica la dirección donde encontrar una tabla de páginas.
- **Bit Usuario/Supervisor:** Indica si la tabla de páginas que referencia puede ser accedida por un usuario.
- **Bit Presente**: Indica si la tabla de páginas está o no cargada en memoria.

Una vez obtenida la dirección de la tabla de páginas debemos de referenciar una de las entradas de esta tabla. Esto se consigue cogiendo los 10 bits siguientes de la dirección lineal y su valor indicará el valor de la entrada. Se puede decir que la estructura del directorio de páginas y las tablas de páginas son equivalentes, excepto que el directorio contiene direcciones de tablas de páginas y las tablas de páginas contiene direcciones de páginas físicas de memoria. Pues bien, a través de la tabla de páginas conseguimos la dirección de una página física de memoria (de 4Kb de longitud). Por último utilizamos los 12 bits menos significativo de la dirección lineal para acceder a una posición concreta dentro de la página física de memoria obtenida anteriormente. Todo este proceso se puede ver resumido en la figura 4.6.

La estructura de este mecanismo de paginado estará formado siempre por un sólo directorio de páginas, varias tablas de páginas y varias páginas físicas. Con este mecanismo se puede tener un programa particionado por toda la memoria y también lo podemos tener cargado parcialmente en la memoria como nos indican los diversos de presencia de páginas en las diversas entradas del directorio y de las tablas de páginas. Utilizando la paginación podemos lograr, por tanto, un sistema de memoria virtual que permita intercambiar páginas de memoria a disco y dar la ilusión al usuario final que se dispone de mucha más memoria que la que realmente hay instalada en el ordenador.

Por último hay que indicar que el procesador necesita en todo momento saber la dirección del directorio de páginas, ya que a partir del directorio comienza todo el proceso de paginado, por lo que guarda su dirección en el registro CR3.

Figura 4.6. Traslación de páginas

- **Protección**

El 80386 posee diversos mecanismos de protección con los que se puede llegar a implementar cualquier tipo de sistema de una forma robusta y eficiente. Vamos a explicar de forma resumida los aspectos más interesantes referente a la protección que ofrece el 80386. Para más información sobre la protección consulte [INTEL386] o [CRAWFORD].

Cada referencia a memoria se comprueba por medio del hardware del 80386 para verificar que efectivamente satisface los criterios de protección. Todas estas comprobaciones se hacen antes de que comience el ciclo de memoria. Cualquier violación evita que este ciclo se empiece y el resultado es una excepción. Por tanto, cualquier acceso a una posición ilegal de memoria desembocará en una excepción.

Una tarea con el máximo nivel de privilegio puede acceder a cualquier segmento de memoria, en cambio, una tarea con el menor nivel de privilegio sólo puede acceder a segmentos de memoria con el mínimo privilegio.

- **Niveles de prioridad**

El 80386 ofrece 4 niveles de prioridad (0,1,2,3) que son asignados a cada uno de los registros de segmento que posee. El nivel de prioridad 0 es el más privilegiado, el 1 el siguiente, así hasta el nivel 3 que posee la menor prioridad. Ante esta situación ofrecida por el 80386, el diseñador de sistemas deberá asignar el nivel de prioridad cero para el sistema operativo, y distribuir los otros tres niveles como desee, a fin de controlar en todo momento los procesos que se ejecutan bajo el sistema. En la figura 4.7 puede observarse una posible asignación de niveles para un sistema operativo bajo un 80386.

Figura 4.7. Niveles de prioridad del 80386

- **Protección a nivel de segmento**

Cómo se comento anteriormente, la memoria la encontraremos dividida en segmentos, accesibles mediante los descriptores de segmento, y cada uno de ellos posee un nivel de prioridad. En cada uno de los descriptores de segmento de la GDT se almacenan toda la información referente a los parámetros de protección para cada segmento.

Las comprobaciones sobre si una tarea tiene permiso a un segmento son realizadas automáticamente por la CPU al mismo tiempo que se cargan los registros de segmento lo que implica un cierto gasto de ciclos de reloj, por ello las aplicaciones bajo el modo protegido irán más lentas que en el modo real.

- **Descriptores de segmento**

En cada uno de los descriptores que conforman la GDT se almacenan ciertos bits que indican diversos parámetros de protección. Cada uno de estos descriptores está formado por 64 bits cuya disposición puede verse en la figura 4.8.

Figura 4.8. Formato de un descriptor de segmento

Cada uno de los parámetros de un descriptor son colocados por medio del software del sistema, al mismo tiempo que se crea un descriptor. Así, en una parte del programa de sistema, debemos de tener un conjunto de instrucciones o datos que formen todo el contenido de la GDT. Los programadores de aplicaciones no deberán de preocuparse en ningún momento de la GDT y de ninguna otras estructuras mencionadas en este capítulo.

Cuando un programa carga un selector en un registro de segmento por ejemplo *mov ds, ax*, no sólo se carga la dirección base del segmento, sino también información referente a la protección del segmento. Así, cada registro de segmento tendrá ciertos bits invisibles para el programador que contienen la información sobre la protección del segmento, con lo que no se tendrá que leer de la GDT en cada acceso al segmento referenciado por el registro de segmento, ya que éste lleva toda la información consigo.

- **Utilización de los campos de un descriptor**

A continuación mostramos como el procesador utiliza cada uno de los bits que conforman cada descriptor de la GDT.

- Como se puede ver en la figura 4.8 el descriptor posee un campo llamado *TIPO* que indica al procesador el tipo de segmento del descriptor. Algunos ejemplos de tipos de segmento se puede ver en la figura 4.9. El tipo se utiliza por el procesador para saber que instrucciones son permitidas bajo ese segmento. Por ejemplo, en un segmento de tipo *código ejecutable* es imposible realizar cualquier escritura sobre él.

Figura 4.9. Tipos de descriptores de puertas y sistemas

- El bit de permiso de escritura en un descriptor de segmento de datos indica si las instrucciones pueden escribir en ese segmento o no.
- El bit de permiso de lectura especifica si los datos de un descriptor de un segmento de datos o código (constantes) pueden ser leídos.
- El campo *LÍMITE* es usado por el procesador para prevenir que los programas accedan fuera del segmento. El valor del límite del segmento viene determinado por el bit *G* (granularidad) que indica la unidad de medida del segmento. Si G=0 entonces el tamaño del segmento es el que contiene realmente el campo LIMITE. Si G=1 entonces el tamaño del segmento es el que contiene el campo LIMITE multiplicado por 4096. Si se intenta acceder a un desplazamiento del segmento mayor que el límite de éste se producirá una excepción de protección general.
- El campo DPL contiene el nivel de prioridad del segmento. Un programa ha de tener mayor o igual privilegio que el DPL del segmento para poder acceder a él. En un sistema operativo se debería de tener todo el núcleo y todas las estructuras que lo forman bajo segmentos con DPL=0, para evitar que programas con menor nivel de privilegio acceda y modifique las estructuras del S.O.

Existen instrucciones privilegiadas que permiten realizar operaciones con los descriptores de la GDT, pero al ser privilegiadas sólo pueden ejecutarse bajo una tarea con el máximo nivel de privilegio (nivel 0). Cualquier intento de ejecutar alguna de estas instrucciones por una tarea con un privilegio menor provocará una excepción por parte del procesador.

- **Protección a nivel de página**

Una vez que tenemos permiso de acceso sobre un segmento, el procesador realiza una comprobación más y comprueba si se puede acceder a la página que queremos acceder dentro del segmento.

- **Formato de una entrada de página**

Como se comento anteriormente, el mecanismo de paginación está compuesto por un directorio y una o más tablas de páginas referenciadas por el directorio. Tanto el directorio como las tablas de páginas está compuesta por una serie de entradas que contienen información acerca de una página de memoria física. Las entradas de páginas son similares a los descriptores de segmentos, pero contiene información diferente. En la figura 4.10 se muestran los campos de una entrada de página. Hay que hacer notar que tantos las entradas del directorio como las de las tablas de páginas son idénticas, la única diferencia es que las entradas del directorio contienen información sobre un tabla de páginas y las entradas de las tablas de páginas contienen información acerca de una página de memoria física.

Figura 4.10. Campos de protección de entrada de tablas de páginas

A continuación mostramos el significado de cada bit de una entrada de página:

- Bit *P* (presente): Este bit es utilizado para indicar si una página se encuentra cargada o no en memoria. La utilización de este bit es especialmente importante para sistemas que gestionen memoria virtual. Un intento de acceder a una página que no está presente en memoria provocará una *falta de página* por parte del procesador. El sistema operativo o el sistema que incluya memoria virtual debería de captar esta excepción y traer la página de disco a memoria, cambiar el bit P de estado y reiniciar la instrucción que provocó la excepción de página.
- Bit *U/S* (usuario/supervisor): Cuando este bit está a 0 indica que cualquier tarea con los niveles de prioridad supervisor (niveles 0, 1 y 2) pueden acceder a esa página. Si este bit está a 1 indica que cualquier tarea puede acceder a esa página, independientemente del nivel de privilegio.
- Bit *R/W* (lectura/escritura): Este bit es utilizado para indicar si la página es de sólo lectura o escritura. Una tarea con un nivel supervisor puede escribir en todas las páginas independiente del contenido de este bit. Las tareas con un nivel de privilegio usuario dependen del contenido de este bit para leer o escribir en la página.

El contenido de estos bits en las entradas del directorio es idéntico al mencionado arriba, lo único en que se diferencia es que en el directorio el valor de cada bit afecta a toda la tabla de páginas que referencia. Por ejemplo, un directorio que tenga el bit R/W a sólo lectura indicará que sólo podremos leer en todas las páginas de la tabla de páginas que apunta el directorio, independientemente del valor de R/W de la tabla de páginas anterior.

- **Multitarea**

Para lograr una multitarea eficiente y protegida, el 80386 emplea varias estructuras de datos especiales. Sin embargo, no usa instrucciones especiales para controlar la multitarea; en su lugar lo que el procesador hace es interpretar las instrucciones de transferencia de control ordinarias (JMP, CALL) de modo diferente cuando éstas se refieren a estructuras de datos especiales. A través de estas estructuras, el 80386 puede realizar rápidas conmutaciones de tarea salvando el contexto de la tarea original.

- **Segmento de estado de tarea**

Toda la información que el procesador necesita para manejar una tarea se almacena en un tipo especial de segmento llamado *segmento de estado de tarea* (TSS). La figura 4.11 muestra el formato de un TSS.

Figura 4.12. Segmento de estado de tarea de 32 bits del 80386

Como puede verse en la figura 4.12, existe una gran cantidad de información que es almacenada en el TSS. Dicha información la podemos dividir en dos grandes grupos:

- Un conjunto dinámico que el procesador actualiza con cada conmutación de tarea. En este conjunto se encuentran los siguientes campos del TSS:
    - Registros generales (EAX, EBX, EDX, ..., EDI).
    - Registros de segmento (CS, ES, SS, DS, FS, GS)
    - Registro de estado e instrucción (EFLAGS, EIP)
    - El selector del TSS de la tarea ejecutada anteriormente
- Un conjunto estático que el procesador lee pero no modifica. En este conjunto se encuentra el resto de campos del TSS.
- Cuando se produce una intercambio de tarea, todo el TSS de la tarea original es cargado por el procesador a fin de recordar el estado de la tarea antes de conmutar. El procesador leerá el TSS de la tarea a la que se le pasa el control y recargará el contenido de sus registros con los valores que ha leído del TSS.
- **Descriptor de TSS**

En la GDT se almacenan también este tipo de descriptores que contienen información acerca de un segmento de memoria que almacena información acerca de un TSS.

El formato de un descriptor de TSS contiene la misma información que cualquier otro descriptor normal de la GDT. El descriptor de TSS contiene por tanto el campo DPL, LIMITE, BASE, etc.

Un procedimiento que tiene acceso a un descriptor de TSS, a través del campo DPL, puede realizar una conmutación de tareas. El tener acceso a dicho descriptor no implica que ese procedimiento pueda leer o modificar el TSS.

- **Registro de tarea**

El procesador contiene un registro llamado TR (registro de tarea) que identifica a la tarea que se está ejecutando apuntando al TSS de esa tarea. Al realizar un intercambio de tarea este registro es recargado a fin de apuntar al TSS de la nueva tarea.

Existen instrucciones como LTR y STR que sirven para cargar y guardar el contenido del registro TR, aunque la mayoría de las veces el registro TR es cargado de forma implícita al realizar un salto sobre un descriptor TSS.

- **Intercambio de tareas**

El 80386 realiza una conmutación de tarea cuando la tarea original realiza un salto o una llamada (JMP o CALL) que remite a un descriptor TSS.

Los pasos siguientes muestran el proceso de un intercambio de tareas:

- Comprobar que la tarea actual tiene permiso de cambiar a la tarea designada. Esta comprobación se realiza a través del campo DPL del descriptor TSS de la nueva tarea.
- Comprobar que el descriptor TSS está marcado como presente y tiene un límite válido (mayor o igual a 103).
- Guardar el estado de la tarea en curso. El procesador guarda en el TSS de la tarea actual el contenido de todos los registros del procesador.
- Cargar el registro de tareas (TR) con el selector del descriptor del TSS de la nueva tarea.

- Cargar el estado de la nueva tarea procedente de su TSS y continuar su ejecución.
- Como puede verse, la conmutación de una tarea a otra es realizada por el 80386 de una forma eficiente y bastante fácil. La única preocupación que tiene el programador es la de definir cuidadosamente los descriptores TSS y definir una área de datos estática para que sea utilizada por el procesador para guardar el estado de cada tarea.
- **Protección en puertos de Entrada y Salida (E/S)**

El 80386 posee instrucciones para acceder a los puertos de E/S, como IN y OUT, pero las comprobaciones que se hacen en éstas, antes de acceder a un puerto específico, son diferentes en el modo real y el modo protegido. Estas instrucciones poseen como operando la dirección de un puerto en el espacio de direcciones de entrada y salida.

Existen dos mecanismos que dan protección a las funciones de entrada y salida:

- El campo IOPL en el registro EFLAGS determina el derecho a usar instrucciones relativas a entrada y salida.
- En el segmento del TSS existe un bit de permiso de entrada y salida para cada puerto de E/S, que especifica e derecho o no de acceder a un puerto específico.

Estos mecanismos de protección sólo operan en el modo protegido y modo virtual 8086, pero no operan en el modo real. Por tanto, en el modo real se puede acceder libremente a cualquier puerto de E/S por cualquier programa, lo que puede suponer una fácil caída del sistema por parte de programas mal intencionados.

- **Protección a través de IOPL**

Las instrucciones que manejan salidas y entradas de datos necesitan ser restringidas, pero también necesitan ser ejecutadas por procedimientos con un nivel de prioridades distinto de cero. Por esta razón, el procesador utiliza el campo IOPL de EFLAGS para dar el nivel de prioridad a cada procedimiento o tarea que se ejecuta. El IOPL está compuesto por dos bits, cuyo valor determina el nivel de prioridad necesario para ejecutar instrucciones de E/S. El contenido de estos dos bits no puede ser cambiado por ningún procedimiento que no posea el nivel de privilegios cero. Un intento de alterar IOPL por un procedimiento de nivel de prioridad distinto de cero, no produce una excepción, sino que simplemente el IOPL queda inalterado.

Vamos a ver como funciona este mecanismo con un ejemplo. Supongamos que tenemos una tarea, tarea A, y ésta se ejecuta en un segmento de código con nivel de privilegios 2, con lo que la tarea A tiene una prioridad 2. También tenemos otra tarea, tarea B, que se ejecuta en un segmento de código de prioridad 1, la prioridad de B es 1. Como cada tarea posee un registro EFLAGS, sus campos IOPL pueden ser distintos. Si tanto la tarea A como la tarea B poseen un IOPL de 1, sólo la tarea B podrá realizar cualquier instrucción de E/S, en cambio, la tarea A tendrá que comprobar el *mapa de bit de permiso de E/S* de su TSS para realizar cualquier instrucción de E/S. A continuación se describe este otro chequeo de acceso a un puerto de E/S.

- **Mapa de bit de permiso de E/S**

Si una tarea no posee mayor o igual prioridad que su campo IOPL para realizar un acceso a un puerto de E/S, el procesador da otra oportunidad para que dicha tarea pueda acceder a un puerto de E/S.

Cómo puede verse en la anterior figura 4.12, el TSS posee un campo de tamaño 8 Kb como máximo llamado *mapa de bit de permiso de E/S*. En este campo se define un bit para puerto de E/S. Por ejemplo el bit 1 indica el tipo de acceso para el puerto 1, el bit 40 indica el acceso para el puerto 40, etc. Un bit con un valor de 0 indica que el acceso para el puerto que especifica está permitido, por el contrario, un bit con un valor de 1 indica que cualquier acceso sobre el puerto que especifica provocará una excepción por parte del procesador.

No es necesario que el mapa de permisos de E/S represente todas las direcciones de entradas y salidas. Las direcciones que no abarca el mapa se tratan como un bit a 1 en el mapa.

Como el mapa de permiso de E/S está en el segmento TSS, tareas diferentes pueden tener diferentes mapas. Así, el sistema operativo puede asignar distintos puertos a cada tarea y conceder a una tarea acceso a un puerto específico cambiando el mapa de permiso de E/S en el TSS de la tarea. Si tenemos una tarea que necesita acceder directamente a la pantalla o manejar el joystick, en caso de ser juego, el sistema operativo puede conceder el acceso directo a los puertos de la pantalla o el joystick a tal tarea.

Hay que destacar sobre todo que este tipo de comprobaciones en el mapa de permiso de E/S se hacen sólo aquellas tareas con un nivel de privilegio menor que el que indica campo IOPL de su registro EFLAGS.

- **Excepciones e interrupciones**

Las excepciones e interrupciones son mecanismos que tiene el 80386 para indicar que se ha producido una violación sobre la seguridad del sistema. Cuando una tarea, por ejemplo, intenta ejecutar una instrucción privilegiada o accede a una página que no está cargada en memoria, el procesador emite una interrupción que puede ser capturada por el sistema operativo y éste último tomará la decisión de qué hacer con esa tarea.

El procesador define un conjunto de identificadores para las interrupciones y excepciones, de forma que el sistema operativo sabe en todo momento el significado de cada excepción/interrupción emitida por el procesador. En la figura 4.13 puede observarse el significado de cada identificador que define el 80386. A partir de las interrupciones que reconoce el procesador, es posible definir más tipos de interrupciones por el sistema operativo.

- **Tabla de descriptores de interrupciones (IDT)**

La tabla de descriptores de interrupciones tiene el mismo formato que la GDT, pero está orientada a segmentos de código que manejan cada una de las interrupciones emitidas por el procesador.

Cuando el procesador, por ejemplo, emite la interrupción 4, el procesador lee la cuarta entrada de la IDT y en ella encontrará la dirección de un segmento donde se encuentra la rutina de manejo para esa interrupción. Esta rutina de manejo de interrupción ha de ser creada por el sistema operativo, y en ella se encuentra las instrucciones que deciden qué hacer con la tarea que provocó esa interrupción.

Las primeras entradas de la IDT están definidas por el procesador y corresponden en orden a las mostradas en la figura 4.13.

Para que el sistema final funcione correctamente, se ha de definir obligatoriamente una IDT y se ha de proporcionar una rutina de manejo de interrupción para cada entrada definida por el procesador. El sistema operativo puede colocar, si desea, nuevas entradas en la IDT que podrá utilizar para ofrecer servicios a las tareas que ejecuta.

Figura 4.13. Asignación de identificadores para interrupciones y excepciones

- **Condiciones de excepción**

En esta sección explicaremos brevemente el significado de cada una de las excepciones que define el 80386 y que han de poseer, como comentamos antes, una entrada en la IDT para que sean tratadas por el sistema operativo.

  - Interrupción 0, *Error de división:* Esta interrupción se da cuando se ejecuta una instrucción de

división cuyo divisor sea 0.

- Interrupción 1, *Excepciones de depurado* (sólo depuradores)
- Interrupción 2, *Interrupción no enmascarable*
- Interrupción 3, *Punto de ruptura:* Utilizada por los depuradores para poner un punto de ruptura en el programa.
- Interrupción 4: *Desbordamiento (Overflow):* Se produce al realizar determinadas operaciones aritméticas donde el resultado no se puede almacenar por los registros del procesador.
- Interrupción 5, *Comprobación de fronteras:* Se produce cuando se ejecuta una instrucción BOUND y encuentra que el operando excede los límites impuestos.
- Interrupción 6, *Código de operación inválido:* Este fallo se produce cuando se intenta ejecutar un código de operación que no existe o no es válido.
- Interrupción 7, *Coprocesador no disponible:* Esta excepción se produce cuando se intenta ejecutar una instrucción del coprocesador y no se dispone de él.
- Interrupción 8, *Fallo doble:* Normalmente, cuando el procesador detecta una excepción mientras trata de llamar al gestor para una excepción previa, ambas excepciones pueden ser tratadas en serie. Sin embargo, si el procesador no puede hacerlo así, señalará una excepción de fallo doble.
- Interrupción 9, *Segmento del coprocesador sobrepasado:* Esta excepción se da en el modo protegido cuando el 80386 detecta una violación de página o segmento mientras se transfieren datos al coprocesador.
- Interrupción 10, *TSS inválida:* Esta interrupción se produce si al intentar conmutar de tarea se detecta un TSS inválido.
- Interrupción 11, *Segmento no presente:* Cuando se intenta acceder a un segmento cuyo bit de presencia indica que el segmento no está en memoria, se produce tal interrupción. El sistema operativo se debería de encargar en este momento de traer el segmento a memoria para que la tarea pueda continuar su ejecución.
- Interrupción 12, *Excepción de pila:* Cuando se intenta cargar el segmento de pila con un segmento no presente se produce tal interrupción. También puede ser dada cuando se intenta acceder fuera de los límites impuestos para la pila.
- Interrupción 13, *Excepción de protección general:* Esta excepción suele ser la que más se da en el sistema, ya que abarca un gran conjunto de situaciones en las que se da esta excepción. Algunas de las situaciones que provoca esta excepción son: sobrepasar los límites de segmento, transferir el control a un segmento no ejecutable, ejecutar una instrucción privilegiada por una tarea con nivel de privilegios distinto de cero, escribir en un segmento de sólo lectura, etc.
- Interrupción 14, *Falta de página:* Esta excepción se da cuando la paginación está activada y se accede a una página que está marcada como no presente.
- Interrupción 15 (reservada).
- Interrupción 16, *Error de coprocesador:* El 80386 informa de esta excepción cuando se detecta una señal del 80287 o del 80387 en el *pin* de entrada ERROR#.
- **Entrando en el Modo Protegido**

  Todas los conceptos sobre el modo protegido para realizar un Multitasker, o al menos DMT, han sido comentado a lo largo de este capítulo. En esta sección comentamos brevemente que es lo que hay que hacer para pasar al procesador del modo real al modo protegido, y poner en funcionamiento todas las características anteriores.

  El 80386 arranca en modo real, donde se chequea la memoria y diversos dispositivos y luego se pasa el control al sistema operativo, en nuestro caso el MS−DOS.

  Para entrar al modo protegido, primero debemos de crear la GDT e IDT y rellenarla correctamente. Si vamos a activar el mecanismo de paginado, como en el caso de DMT, debemos de crear también el directorio de páginas y las correspondientes tablas de páginas. Una vez realizado esto, ya podemos conmutar al modo protegido. Para ello, activaremos el bit PE de registro CR0, con lo que el

procesador ya estará en modo protegido y luego activaremos el bit PG en el mismo registro para activar la paginación.

Así de fácil es entrar en el modo protegido pero recordad, ahora ya no están disponibles los servicios que ofrece el MS−DOS ni la BIOS, por lo que las instrucciones para solicitar un servicio, como *int 21h*, provocará una excepción de protección general por parte del 80386.

16

REGISTROS GENERALES

31 23 15 7 0

EAX

EBX

ECX

EDX

ESP

EBP

ESI

EDI

AX

AH AL

BX

BH BL

CX

CH CL

DX

DH DL

SP

BP

SI

DI

REGISTRO EFLAGS

31 23 15 7 0

VM RF O NT IOPLOF DF IF TF SF ZF 0 AF 0 PF 0 CF 1CF

Reservado por INTEL

15 0 31 0

DIRECCIÓN

LÓGICA

Paginación Desactivada

DIRECCIÓN

LINEAL

DIRECCIÓN

FÍSICA

Desplazamiento

Selector

TRASLACIÓN DE

SEGMENTO

PG?

DIR PAGINA DESPLAZAMIENTO

TRASLACIÓN DE

PÁGINA

DIRECCIÓN

LÓGICA

TABLA DE

DESCRIPTORES

DIRECCIÓN

BASE

DIRECCIÓN

LINEAL

DESPLAZAMIENTO

SELECTOR

DESCRIP.

DE SEGMENTO

+

DIR PÁGINA DESPLAZAMIENTO

31 22 21 12 11 0

DIR PÁGINA DESPLAZAMIENTO

DIRECTORIO TABLA

DE PAGINAS DE PAGINAS

DIR PAGINA DESPLAZAMIENTO

DIRECCIÓN

FÍSICA

ENTRADA

DE PAGINA

ENTRADA

DE DIR.

CR3

A

Aplicaciones

Extensiones para

el cliente

Servicios

Nivel

2

Nivel

3

Niv.

1

Núcleo

Niv 0

32 23 15 7 0

BASE 23..15

TIPO

DPL

P

LÍMITE

19..15

A

V

L

BASE 31..24 0 G B

LIMITE DEL SEGMENTO 15..0

BASE DEL SEGMENTO 15..0

Código	**Tipo de segmento o puerta**
0	reservado
1	Disponible TSS del 286
2	LDT
3	TSS del 286 ocupado
4	Puerta de llamada
5	Puerta de tarea
6	Puerta de interrupción del 286

7	Puerta de trampa del 286
8	reservado
9	Disponible TSS del 386
A	reservado
B	TSS del 386 ocupado
C	Puerta de llamada del 386
D	reservado
E	Puerta de interrupción del 386
F	Puerta de trampa del 386

U R

Dirección de página 31..12 Disp. 0 0 D A 0 0 / / P

S W

31 15 7 0

BASE DEL MAPA DE E/S	0 0 0 0 0 0 0 0 0 0	T
0 0 0 0 0 0 0 0 0 0 0 0	LDT	
0 0 0 0 0 0 0 0 0 0 0 0	GS	
0 0 0 0 0 0 0 0 0 0 0 0	FS	
0 0 0 0 0 0 0 0 0 0 0 0	DS	
0 0 0 0 0 0 0 0 0 0 0 0	SS	
0 0 0 0 0 0 0 0 0 0 0 0	CS	
0 0 0 0 0 0 0 0 0 0 0 0	ES	
EDI		
ESI		
EBP		
ESP		
EBX		
EDX		
ECX		
EAX		
EFLAGS		
Puntero Instrucción (IP)		
CR3		
0 0 0 0 0 0 0 0 0 0 0 0	SS2	
ESP2		
0 0 0 0 0 0 0 0 0 0 0 0	SS1	
ESP1		
0 0 0 0 0 0 0 0 0 0 0 0	SS0	
ESP0		
0 0 0 0 0 0 0 0 0 0 0 0	Unión al anterior TSS	

Identificador	**Descripción**
0	Error de división
1	Excepción de depurado
2	Interrupción no enmascarable
3	Punto de ruptura
4	Desbordamiento
5	Comprobación de fronteras
6	Código de operación inválido
7	Coprocesador no disponible
8	Fallo doble
9	reservado
10	TSS inválido
11	Segmento no presente
12	Excepción de pila
13	Falta de protección general
14	Falta de página
15	reservado
16	Error de coprocesador
17−31	reservado
32−255	Disponible para usuario