FUNCIONES ESPECIALES, ENTORNOS DE TRABAJO Y COMPATIBILIDAD

19

19.1 CAPACIDAD DE AUTOCOMPROBACIÓN.

El Pentium es capaz de auto chequearse hasta el 70% del silicio que posee, comprendiendo toda la memoria ROM de control y la mayoría de la lógica no aleatoria. Este auto chequeo se produce mediante la operación BIST (Built-In Self-Test) del Pentium. Para que esta operación pueda ejecutarse se debe conectar:

- La alimentación.
- La señal de entrada Reset debe subir a nivel alto.
- La patita Busy # debe de conectarse a Tierra.

El auto chequeo es un programa de autocomprobación cuyo tiempo de ejecución varía según el tipo de procesador.

Tras ejecutarse dicho programa deja en EAX un código binario que se trata del resultado del auto chequeo. Si todos los bits son 0 entonces la operación ha sido ejecutada con éxito, si por el contrario se encuentra un valor distinto de cero en el registro significa que se ha dado un fallo en el procesador y hay que acudir a las características técnicas del fabricante para saber que anomalía se ha producido.

En EDX queda depositada una información sobre las características técnicas del Pentium que se ha chequeado, así como su número de versión.

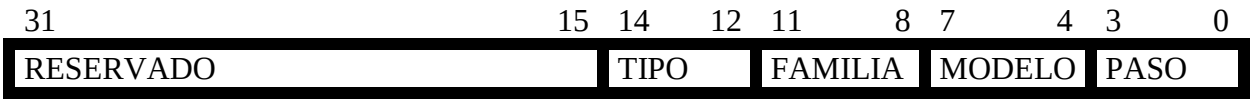

Figura 19.1-Formato del registro EDX

19.1.1 Comprobación de la TLB

La TLB es una memoria tipo CAM (acceso por contenido) que está realizada con una memoria caché ultrarrápida, dónde hay etiquetas y datos. En la etiqueta pone una dirección lineal y en los datos la dirección física. La TLB guarda las 256 últimas traducciones de las últimas páginas utilizadas.

La arquitectura interna del Pentium dispone de 2 registros TR6 y TR7 mediante los cuales se puede leer y escribir cualquier posición de la TLB.

Para realizar el auto chequeo se precisa un programa en lenguaje Ensamblador, que controla un mapa de pruebas. También es recomendable desactivar la paginación para impedir que los resultados de las pruebas eliminen a los datos obtenidos del manejo de la memoria con la paginación.

TR6 actúa como registro de comandos para el control de las pruebas, mientras que TR7 es el que se carga con los resultados de las pruebas, una vez efectuadas.

19.2 SOPORTE A LA DEPURACIÓN

La depuración es la operación que se realiza para comprobar y ajustar el funcionamiento, tanto del equipo físico como del sistema lógico.

El Pentium dispone de tres recursos eficaces para el soporte de la depuración:

- **Punto de Parada por Software**: Se consigue detener el procesamiento de un programa en un punto determinado del mismo, introduciendo allí una instrucción de interrupción. Se trata de INT3, que está codificada en un sólo byte y su ejecución origina interrupción, que es atendida por la entrada 3 de la IDT. Allí, el programador del sistema puede situar la referencia a un programa que visualice los elementos deseados de la arquitectura del procesador y del estado del sistema.

- **Excepción paso a paso:** Cuando se habilita el señalizador TF del registro de EFLAGS (TF=1), cada vez que se ejecuta una instrucción se produce una excepción que es atendida por la entrada 1 de la IDT. La excepción pasa a paso almacena el registro de señalizadores en la pila, con el bit TF=1. Después, se desactiva TF para poder procesar con normalidad la rutina de servicio a dicha excepción, que al concluir con la instrucción IRET, se vuelve a colocar (desapilar) el registro de señalizadores como estaba, con TF=1, transfiriéndose el control a la siguiente instrucción del programa principal.

- **Registros de Depuración (DR0-DR7):** Mediante estos registros se pueden programar cuatro puntos de parada independientes. Los registros DR0-DR3 contienen la dirección lineal de los cuatro posibles puntos de parada. DR7 sirve para especificar la condición que se desea detectar par originar la parada del procesamiento. También se puede determinar la granularidad, así como enmascarar cada condición por separado y definir la entrada correspondiente: local para una tarea, o bien, global para todas las que hay en el sistema. DR6 guarda la información necesaria para conocer la causa que ha provocado una parada, al detectarse una condición de punto de parada.

19.3 EL RESET Ó LA INICIALIZACIÓN

Manteniendo activa la patilla RSET durante un cierto tiempo que viene determinado y que no es el mismo para cada Pentium, el procesador del sistema comienza un hardware de inicialización del procesador (conocido como RESET del hardware) y una autocomprobación opcional llamada BIST.

Cuando se activa la patita Reset se deja a los registros en un estado conocido y coloca al procesador en modo Real. Borra y limpia la caché interna, la TLB y también la BTB(otra caché especial en la que se guarda el resultado de los últimos 256 saltos condicionales, y que se utiliza para prever el resultado de un salto condicional. En un 97% de los casos acierta, si hay fallo hay que retirar las instrucciones que hemos metido en el cauce y limpiar los registros).

La activación de la patilla INIT# en el procesador tiene un efecto similar al RESET del hardware. La diferencia está en que durante un INIT# los estados de las cachés internas y e FPU permanecen inalterados. La activación del INIT# permite cambiar de modo protegido a modo real manteniendo el contenido de las cahés internas.

El estado de los registros es el siguiente:

EFLAGS	0000 0002 hex	
EIP	0000 FFF0 hex	
CS	Base	FFFF 0000
	Límite	FFFF
EAX=EBX=ECX=EDX=EDI=ESI=EBP=ESP	0	
CR0	6000 0010 hex	

1ª Conclusión: Si el Registro CR0 tiene un 0 en el bit de menos peso , el bit PE se encontraría a 0 por lo que el Pentium siempre comenzaría trabajando en modo real . Por otro lado también es 0 el bit de mas peso , que se trata del bit PG de manera que al comenzar a trabajar el Pentium lo hace con la paginación desactivada.
(Dibujo del registro cr0)

2ª Conclusión: En modo Real la Memoria Principal sólo utiliza el primer Mbyte de la memoria física. Por lo que en Modo Real las 12 líneas de menos peso no son utilizadas, sin embargo la primera dirección que vaya después del Reset entra por las 32 líneas.

(Dibujo de la Memoria Principal)

La primera instrucción que se ejecuta después de un Reset se hallará sumando la base más el EIP de modo que queda la dirección física : FFFF FFF0 en hexadecimal.

Base:	FFFF 0000
EIP :	FFF0
Dirección física:	FFFF FFF0

La 1ª dirección no está dentro del mapa de memoria que le corresponde al modo Real por lo que en esa posición hay una instrucción de salto a la 000F FFF0 que se trata de la 1ª dirección real de nuestro programa.

3ª Conclusión: Nada mas empezar a trabajar se puede generar una interrupción ó una excepción, por lo tanto antes de empezar a trabajar la IDT tiene que estar residente en memoria principal. Por defecto, la dirección física de la IDT es 0000 0000 hexadecimal.

19.4 MODO REAL

Este modo de operación implementa el entorno de programación del procesador Intel 8086 , imprimiéndole mas velocidad a la hora de ejecutar programas.

En Modo Real se empieza a trabajar siempre que hay un Reset ó cuando se conecta por 1ª vez la alimentación.

Características del entorno de ejecución del Modo Real:

1-La Memoria Principal sólo es de 1Mbyte. Este espacio está dividido en segmentos , cada uno de los cuales puede tener una longitud de 64 Kbytes .

2-El bus de direcciones sólo usa las 20 líneas de menos peso.

3-Es monotarea .

4-Los 8 registros de propósito general(AX, BX, CX , DX, SI, DI, SP y BP) son de 16 bits.

5-El puntero de instrucciones (IP) del 8086 está manejado en los 16 bits de menos peso del registro EIP.

6-El registro de 16 bits FLAGS contiene los flags de control y estado. (Este registro está mapeado en los 16 bits menos significantes del registro de 32 bits EFLAGS).

7-La tabla IDT es de 256 entradas, con entradas de 8 bytes y en cada una hay un vector de interrupciones que apunta a la tabal de interrupciones que tiene entradas de 4 bytes.

(Dibujo de la IDT en Modo Real)

19.5 MODO PROTEGIDO

Es el modo natural que tiene de trabajar el Pentium, los registros tienen ya una extensión de 32 bits, se trabaja con multitarea, se puede poner en marcha la paginación y funciona el modo de protección.

Estando en Modo Real para pasar a Modo Protegido se pone el flag PE del CR0 a 1, que habilita el mecanismo de protección de segmentos, pasando automáticamente a Modo Protegido.

Al pasar a Modo Protegido hay unas cuántas cosas que hay que tener en cuenta:

- IF=1 (Interrupciones mascarables prohibidas).

- La IDT en Modo Protegido es completamente distinta de la IDT en Modo Real. La IDT del Modo Protegido no contiene vector de interrupciones , contiene descriptor de Interrupciones. La IDT en Modo Protegido es completamente distinta en tamaño y contenido de la IDT en Modo Real. La IDT cuando entramos en Modo Protegido tiene que estar ya cargada, de manera que justo antes de entrar en Modo Protegido tenemos que crear la IDT en memoria Principal. La IDT debe estar situada en el primer kilobyte de la memoria física.

- Estando aún en Modo Real, hay que construir una imagen básica de la GDT e inicializar el registro GDTR.

Hay que inicializar el segmento de pila, direccionando una zona de RAM. Simplemente, será preciso cargar los valores SS y ESP con los correspondientes a la zona deseada, lo cual se hace a base de instrucciones MOV.

Entonces hay una serie de pasos que hay que dar antes de pasar a Modo Protegido:

1º Deshabilitar las interrupciones . Una instrucción CLI (IF=0) deshabilita las interrupciones mascarables.

2º Instalar en memoria la GDT y apuntarla mediante LGDTR (instrucción que nos apunta a la GDT.

3º Ejecutar la instrucción MOV CR0 que pone el flag Pe a1 (Meter un 1 en CR0, por lo que cargo en EAX 0000 0001 hex y de ahí movemos a CR0)

4º Hacer una instrucción Jump. Cuando estamos en Modo Real hay un elemento que se denomina Cola de Prebúsqueda que tiene cargados códigos de Instrucciones de Modo Real, por lo que antes de pasar a Modo Protegido hay que limpiar la pila. Esto se hace con la Jump que elimina las instrucciones que quedan por detrás de la posición a la que salto.

5º Ejecutar la instrucción LLDT (Load LDT). Carga en LDTR la dirección de la LDT de la tarea que hay en curso.

6º Ejecutar la instrucción LTR. Carga el registro de tareas con un selector de segmentos a la tarea inicial del modo protegido o a un área escribible de memoria que pueda ser usada para cargar la información del TSS en un cambio de tarea.

7º Actualizar los registros de segmentos para que apunten a los segmentos de Modo Protegido.

8º Ejecutar la instrucción LIDT para cargar el registro IDTR con la dirección y el límite del IDT del modo Protegido.

9º Ejecutar la instrucción STI (Permitimos las interrupciones mascarables. Hasta ahora no hemos permitido las interrupciones de los periféricos.)

19.6 MODO VIRTUAL 86

Se trata de un modo que es mezcla del Modo Real y del Modo Protegido. En el Modo Protegido, el procesador puede trabajar en un ambiente conocido como Modo Virtual_86 o modo de tarea especial. Este modo permite al procesador ejecutar software del 8086 en un entorno protegido y multitarea.

Cuando el sistema operativo cambia a una tarea en Modo Virtual_86, el procesador simula un procesador 8086 donde el entorno de ejecución es igual al Modo Real con la diferencia de que este modo puede utilizar servicios del Modo Protegido.

Un aspecto a tener en cuenta es que el Modo Virtual_86 puede ejecutar algunas tareas que son del Modo Real y como sabemos este modo trabaja con direcciones de 16 bits, mientras que el Modo Virtual_86 trabaja con direcciones de 32 bits. Por esto el Modo Virtual_86 tendrá que convertir las direcciones de 16 bits a 32 bits.

Para entrar en este modo, el procesador tiene que estar dentro de cualquiera de estas situaciones:

- Cuando el flag VM del registro E_FLAGS es puesto a 1 en la TSS de la tarea.
- Cuando se retorna de una interrupción o excepción en modo protegido, cuando el flag VM es puesto a 1 en el registro EFLAGS

El procesador sólo podrá abandonar el Modo Virtual_86 mediante una interrupción o una excepción.

El procesador sólo testeará el flag VM en los siguientes casos:

- Cuando se cargan los registros del segmento, para determinar si hay que usar la traducción de dirección en estilo 8086.
- Cuando decodifica las instrucciones para determinar qué instrucciones son las no válidas en Modo Virtual_86.

Al comprobar instrucciones privilegiadas, en accesos a páginas, o cuando se realiza cualquier otra comprobación de permiso.

19.7 MODO MANEJO DEL SISTEMA (SSM)

En este modo, el procesador proporciona un sistema operativo que es transparente para el programador y que implementa dos funciones muy importantes:

- Mejora de la seguridad de todo el sistema.
- Un sistema de control de la alimentación. Mediante este sistema controla el consumo de energía y realiza un consumo lo más óptimo posible.

Cuando se pasa a Modo SMM, el procesador aísla completamente un espacio de memoria reservado para él donde se conmuta mientras guarda todo el contexto de la tarea que va a ejecutar. Es entonces cuando ejecuta el código específico del SMM. Cuando se retorna del modo SMM, el procesador vuelve a su estado anterior a la interrupción del manejo del sistema.

Para pasar de cualquiera de los modos a modo SMM, habrá que activar por hardware una patita del Pentium. Esa patita es la SMI, que se activa por nivel bajo (SMI#). La interrupción que provoca esta patita tiene mayor prioridad que las interrupciones externas o las excepciones de depuración. Lo primero que realizara el procesador será salvar el estado en el que se encuentra el procesador y cambiar a un entorno operativo reservado por él y contenido en l a RAM de manejo de sistema.

Mientras el procesador esté en este modo, ejecutará el código del manipulador del SMI para realizar ciertas operaciones como desactivar unidades de disco o monitores, poniendo todo el sistema en un modo de reposo. Cuando el manipulador SMI a completado sus operaciones, ejecuta una instrucción de salida (RSM) que hace que el procesador cargue de nuevo el contexto de la tarea que estaba ejecutando antes de iniciar este modo y por último vuelve al Modo Real o al Modo Protegido dejando de ejecutar la aplicación de interrupción o el programa del sistema operativo.

Para pasar a SMM a los otros tres modos existentes, existen tres formas:

- RSM + PE = 0→ Modo Real.
- RSM + PE = 0→ Modo Protegido.
- RSM + PE = 1 + VM = 1 → Modo Virtual 86.

El modo SMM es similar al Modo Real en que no hay niveles de privilegio ni mapas de direcciones. Un programa en modo SMM puede tener hasta 4Gbytes de memoria y puede ejecutar todas las instrucciones del sistema así como todas las instrucciones de E/S.