CONMUTACIÓN DE TAREAS 13

13.1- IMPLANTACIÓN DE LA MULTITAREA

Una tarea es un conjunto de segmentos u objetos ubicados en memoria, que, al ser procesados por la CPU, producen un resultado final, que es el objetivo de la tarea.

Un sistema multitarea permite al procesador atender simultáneamente a varias tareas independientes, de tal forma que, cuando abandona una tarea, pasa al procesamiento de otra. Si la CPU es muy rápida y la operación de conmutar tareas dura poco, parece como si cada tarea tuviese permanentemente a su disposición la CPU.

Para un programador, la multitarea es el procesamiento discontinuo de varias tareas que aparentemente se ejecutan de forma paralela.

Una conmutación de tarea consiste en abandonar el procesamiento de la tarea en curso (vieja) para reanudar otra tarea (nueva).

Para reanudar una tarea en cualquier momento, hay que disponer de algún objeto que almacene el estado completo del procesador cuando abandonó anteriormente dicha tarea. Dicho estado se compone, fundamentalmente, del contenido de la mayor parte de los registros de la CPU, y recibe el nombre de "contexto". Debe contener la LDT específica, el registro que apunta al Directorio de Páginas, los punteros de pila para los diversos niveles de privilegio, los restantes registros del procesador, parámetros del S.O., etc.

Cada tarea tiene guardado el contexto de la CPU que le corresponde en un segmento que se denomina Segmento de Estado de la Tarea, TSS.

El procesador tiene cuatro registros para la gestión de memoria:

- GDTR (*global descriptor table register*): contiene los 32 bits de la dirección base y los 16 bits del límite de tabla de la GDT.
- LDTR (*local descriptor table register*): contiene los 16 bits del selector de segmento, 32 bits de la dirección base, 16 bits del límite de segmento y atributos del LDT.
- IDTR (*interrupt descriptor table register*): contiene los 32 bits de dirección base y 16 bits del límite de tabla del IDT (tabla de interrupciones).
- TR (*task register*): contiene 16 bits del selector de segmento, 32 bits de la dirección base, 16 bits del límite de segmento y atributos del TSS (segmento del estado-tarea) de la tarea actual.

13.2- SEGMENTO DE ESTADO DE LA TAREA, TSS.

El segmento TSS, como es específico para cada tarea, se define mediante valores de selector almacenado en los registros de segmento del sistema.

En la figura 13.1 se muestra la estructura mínima del TSS, para una CPU de 32 bits, compatible con tareas de procesadores de 16 bits. Además, puede contener otras informaciones auxiliares complementarias, que precisa el S.O. para reanudar la tarea.

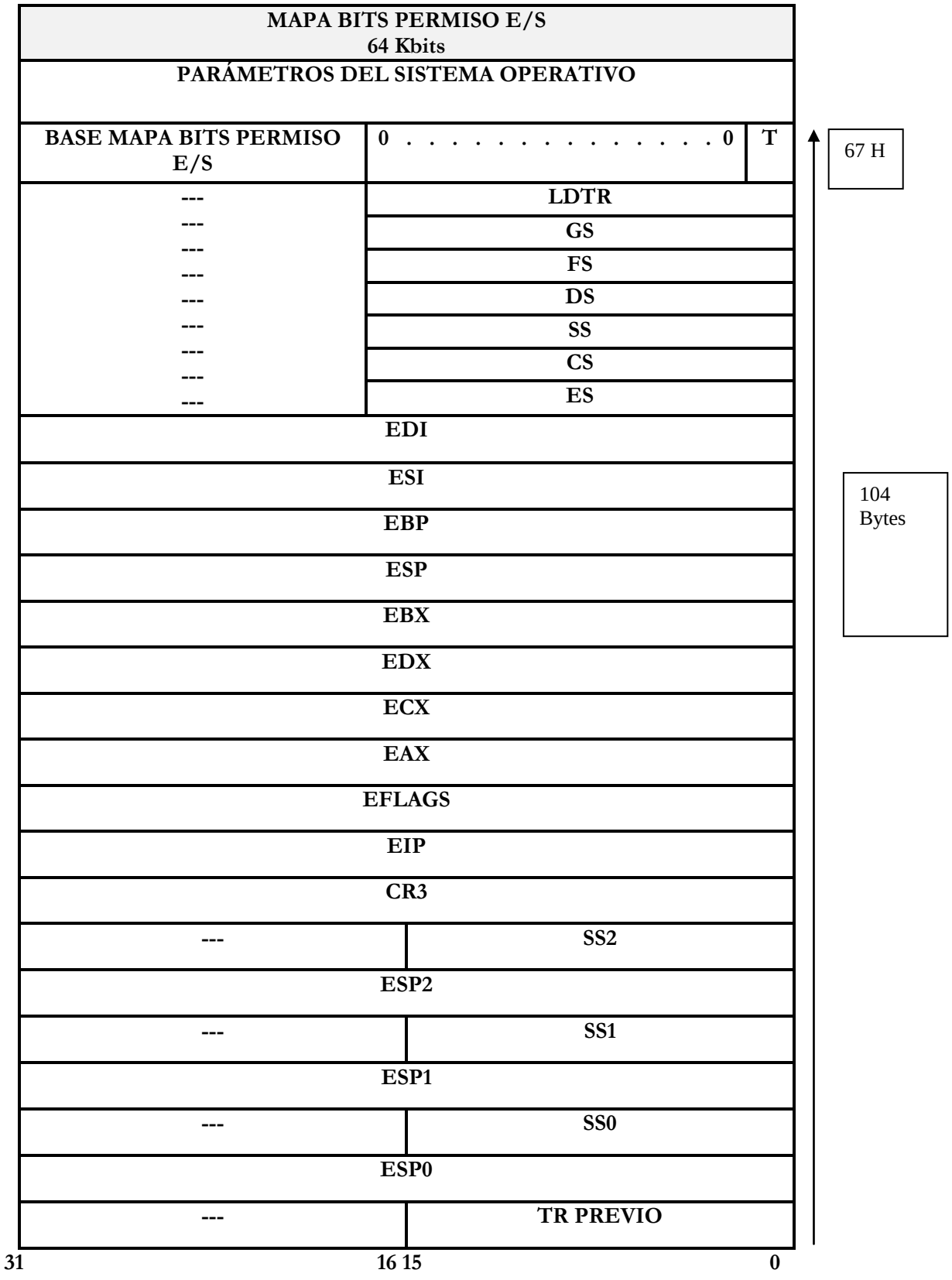


Figura 13.1. Estructura del segmento TSS. Dispone de una capacidad mínima de 104 bytes, que se precisan para guardar el contexto de la CPU.

Los campos son:

1. Registros de propósito general. El estado de los registros EAX, ECX, EDX, EBX, ESP, EBP, ESI y EDI antes de la conmutación de tarea.

2. Selectores de segmento. Los selectores de segmento almacenados en los registros ES, CS, SS, DS, FS y GS previos a la conmutación de tarea.

3. El registro EFLAGS. Estado del registro EFLAGS antes de la conmutación de tarea.

4. Campo EIP. Estado de EIP antes de la conmutación.

5. El TR previo. El selector de segmento TSS de la tarea previa. (Actualiza la conmutación de tarea que comenzó por una instrucción CALL, una interrupción, o una excepción). Este campo permite la conmutación a la tarea previa mediante una instrucción IRET.

6. Selector de segmento LDTR.

7. Registro de Control CR3. Contiene la base del directorio de páginas a ser usada por la tarea. El registro de control CR3 también es conocido como el registro base del directorio de páginas.

8. Punteros de la pila de nivel de privilegio 0, 1 y 2. Los punteros de la pila consisten en una dirección lógica compuesta por el selector de segmento para el segmento de pila (SS0, SS1, y SS2) y un desplazamiento en la pila (ESP0, ESP1, y ESP2). Los valores en estos campos son estáticos para una tarea particular, teniendo en cuenta que los registros SS y ESP cambiarán si la conmutación de la pila ocurre dentro de la tarea.

9. Flag T (Flag de depuración) (byte 100, bit 0). Cuando esta activo y se produce una conmutación de tarea, el flag T causa en la CPU una excepción de depuración.

10. Base del mapa de bits de E/S. Contiene el desplazamiento de 16 bits que hay que sumar a 67H para apuntar al mapa de bits de permiso de E/S.

11. Parámetros del S.O.

En la figura 13.1 no están incluidos en el contexto de la CPU, los valores que referencian al segmento de pila de nivel de privilegio 3, o sea, el valor de los registros SS3 y ESP3. La razón es la siguiente:

1. Si cuando se está trabajando en el nivel de privilegio 3, se guarda el contexto del procesador, los valores SS3 y ESP3 son los que están en curso, y por tanto, están salvados en los registros generales SS y ESP.

2. Si el momento en el que se salva el contexto de la CPU tiene un nivel de privilegio diferente del 3, no es necesario conocer los valores de SS3 y ESP3, pues resulta imposible rebajar el nivel de privilegio del código.

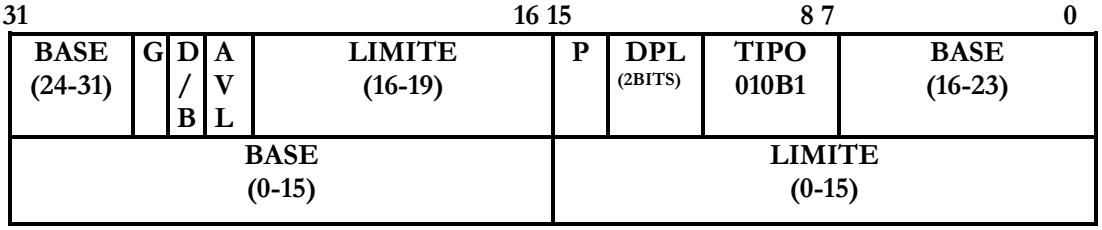
La conmutación de tarea se llevará a cabo de una manera más eficiente si las páginas que contienen estas estructuras también están presentes en la memoria antes de que la conmutación comience.

La CPU hace referencia al TSS por medio del registro TR, Registro de Tarea, que actúa como un selector de la GDT, seleccionando uno de sus descriptores que corresponden con TSS. Se trata de uno de los llamados segmentos del sistema.

El flag BUSY (B) en el campo TIPO indica si la tarea está ocupada. El campo TIPO de los atributos tienen el código 010B1. El bit B indica que la tarea está libre (B = 0) u ocupada (B = 1) (figura 13.2).

El procesador usa el flag BUSY para detectar una llamada a una tarea cuya ejecución se ha interrumpido. Para asegurar que hay sólo un flag BUSY asociado con una tarea, cada TSS debe tener solamente un descriptor TSS que apunte a él.

DESCRIPTOR DE UN SEGMENTO DE ESTADO DE TAREA.



AVL	Libre para ser usado por el software del sistema.
B	Flag BUSY.
BASE	Base del segmento.
DPL	Nivel de privilegio del descriptor.
G	Granularidad.
D/G	Defecto/Grande.
LIMITE	Límite.
P	Presencia
TIPO	010B1

Figura 13.2. Estructura de un descriptor de TSS situado en la GDT. Si el campo TIPO tiene de valor 9, se trata de una tarea libre y si es B, es una tarea ocupada.

La base, el límite, los campos DPL, la granularidad y el flag de presencia tienen funciones similares de uso en descriptores de segmentos de datos. Cuando el flag G es 0 en un descriptor de un TSS de 32 bits, el límite debe tener un valor igual o mayor que 67H, un byte menos del tamaño mínimo de un TSS.

El hecho de intentar conmutar a una tarea cuyo descriptor TSS tiene un límite menor de 67H genera una excepción, TSS no válido. Se requiere un límite más grande si el mapa de bits de permiso de E/S es incluido en el TSS. Se requeriría un límite aun más grande si el sistema operativo guardara datos adicionales en el TSS. La CPU en una conmutación de tarea no verifica límites mayores de 67H, sin embargo, lo hace cuando accede al mapa de bits de permiso de E/S o redirecciona la interrupción del mapa.

Cualquier programa o proceso que tenga acceso a un descriptor TSS (cuando su CPL es numéricamente igual o menor al DPL del descriptor de TSS) puede avisar a la tarea con una CALL o una JMP.

En la mayoría de los sistemas, el DPL de los descriptores TSS deben ponerse con valores menores de 3 para que solamente el software con privilegios pueda realizar la conmutación de tarea. Sin embargo, en las aplicaciones multitarea, el DPL de algunos descriptores TSS puede ponerse a 3 para permitir la conmutación de tarea del nivel de privilegio de la aplicación.

13.3- EL REGISTRO DE TAREA, TR.

El registro de tarea es un segmento de 16 bits que apunta de forma indirecta al TSS. Con esos 16 bits apunta a un descriptor de la GDT del cual se obtiene la base y el límite del TSS, que se cargan en un registro caché oculto asociado al TR. Guardar estos valores en un registro caché hace que la ejecución de la tarea sea más eficiente, porque el procesador no necesita ir a la memoria a por estos valores para referirse al TSS de la tarea actual.

Las instrucciones LTR (carga del registro TR) y STR (almacenamiento de TR) cargan y leen la parte visible del registro de la tarea. La instrucción LTR carga un selector de segmento (operando fuente) en el registro de tarea que apunta a un descriptor TSS en la GDT, y entonces se carga la parte invisible del registro de tarea con información del descriptor TSS. Esta instrucción es una instrucción privilegiada que sólo puede ejecutarse cuando el CPL es 0. La instrucción LTR generalmente se usa durante la inicialización del sistema para poner un valor inicial en el registro de tarea. Después, los contenidos del registro de tarea son cambiados implícitamente cuando ocurre una conmutación de tarea.

La instrucción STR (almacenamiento de TR) guarda la parte visible del registro de tarea en un registro de propósito general o en memoria. Esta instrucción puede ejecutarse desde cualquier nivel de privilegio para identificar la tarea que está corriendo actualmente, sin embargo, normalmente se usada sólo por el software del sistema operativo. Al arrancar o resetear el procesador, el selector de segmento y la base un cargan con el valor por defecto (predefinido 0) y el límite es puesto a FFFFH.

En la figura 13.3 se representa gráficamente el mecanismo que permite a la CPU direccionar al TSS a través del Registro de Tarea (TR).

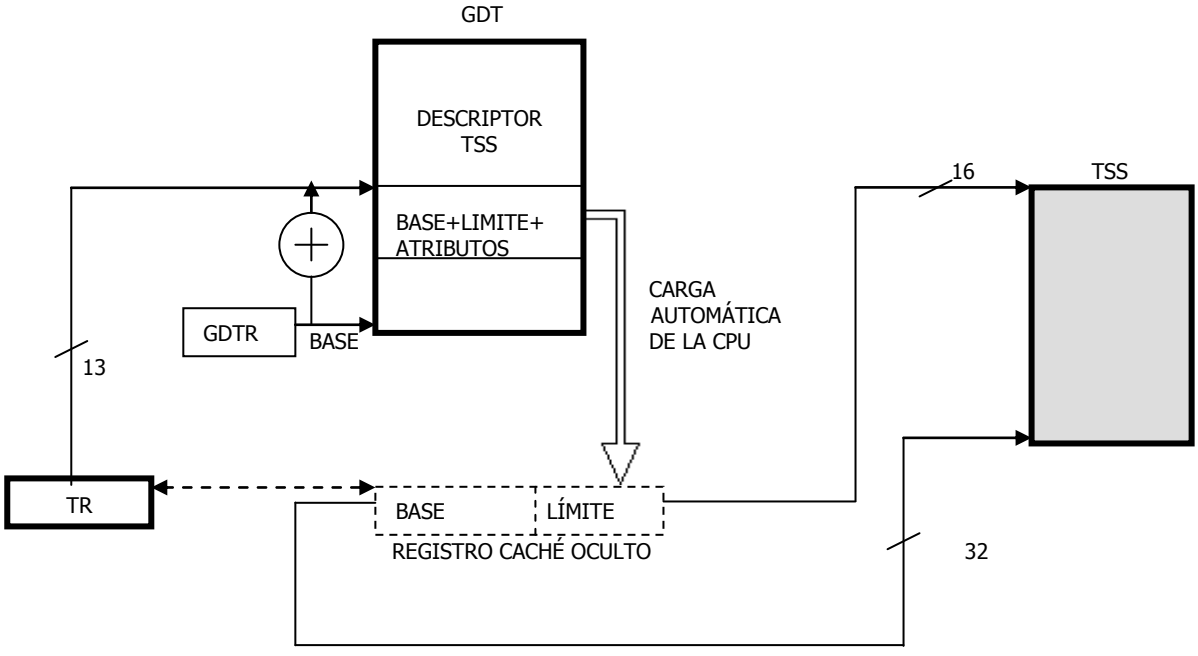


Figura 13.3. Descripción gráfica del mecanismo de direccionamiento del segmento TSS.

13.4- CONMUTACIÓN DE TAREA

El Pentium no dispone de instrucciones específicas para provocar la conmutación de tarea. Utiliza JMP y CALL referidos al nuevo TR desde segmentos de código que deben poseer el mismo nivel de privilegio que el TSS, lo cual supone la restricción de poder realizar únicamente la conmutación desde el máximo nivel de privilegio. Como se verá más adelante, es posible realizar una conmutación de tarea mediante una interrupción.

La CPU transfiere la ejecución a otra tarea en cualquiera de estos cuatro casos:

1. El programa actual, la tarea, o el proceso ejecuta una JMP o instrucción CALL a un descriptor de TSS en la GDT.

2. El programa actual, la tarea, o el proceso ejecuta un JMP o instrucción CALL a un descriptor de puerta de tarea en la GDT o la LDT actual.

3. Una interrupción o el vector de la excepción apunta a un descriptor de la puerta de tarea en el IDT.

4. La tarea actual ejecuta un IRET cuando el flag NT situado en el registro EFLAGS se pone a uno.

La conmutación de tarea se compone de cuatro etapas fundamentales:

1. En la tarea vieja se ejecuta una instrucción JMP o CALL con el TR_NUEVO. Las dos instrucciones hacen lo mismo, la diferencia está en la CALL que incluye anidamiento de tarea.

2. Se carga en TR el valor del TR_NUEVO.

3. Se salva el contexto de la CPU en el TSS_VIEJO.

4. Se carga el TSS_NUEVO como contexto de la CPU.

En la figura 13.4 se muestra gráficamente el mecanismo simplificado de la conmutación de tarea.

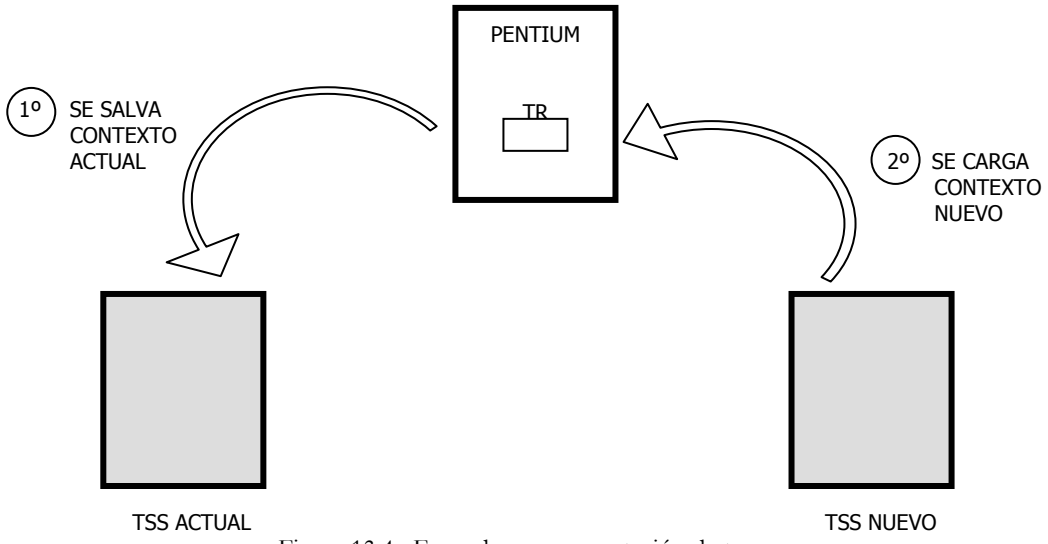


Figura 13.4. Fases de una conmutación de tarea.

Las instrucciones JMP, CALL, e IRET, así como las interrupciones y excepciones, son mecanismos generales para redirecionar un programa. La referencia a un descriptor TSS o una puerta de tarea (cuando llama o salta a una tarea) o el estado del flag NT (al ejecutar una instrucción IRET) determina si ocurre una conmutación de tarea. La CPU realiza las siguientes operaciones al cambiar a una nueva tarea:

1. Obtiene el selector del TSS nuevo del operando de la instrucción, de una puerta de tarea, o del TSS previo, para una conmutación de tarea que comienza con una instrucción IRET.

2. Comprueba que la tarea actual (vieja) permite conmutar a la tarea nueva. de. El CPL de la tarea actual (vieja) y el RPL del selector de segmento para la tarea nueva deben ser menor o igual al DPL del descriptor TSS o puerta de tarea a la que nos referimos.

3. Comprueba que el descriptor TSS de la tarea nueva está presente y tiene el límite válido (mayor que o igual a 67H) , que la nueva tarea está disponible y que el TSS actual (viejo), el TSS nuevo, y todos los descriptores de segmento usados en la conmutación de tarea están dentro la memoria del sistema.

4. Guarda el estado de la tarea actual (vieja) en su TSS asociado. La CPU encuentra la base del TSS actual mediante el registro de tarea y entonces copia en el TSS el estado de los siguientes registros: registros de propósito general, selectores de los registros de segmento, la imagen del registro EFLAGS, y el registro de instrucción del puntero (EIP).

5. Carga el registro de tarea con el selector de segmento y descriptor para el TSS de la tarea nueva.

6. Carga el estado de la tarea nueva en la CPU. Cualquier error asociado con la carga y verificación del descriptor de segmento ocurre en el contexto de la tarea nueva. La información del estado de la tarea incluye el registro LDTR, registro de control CR3, registro EFLAGS, registro EIP, registros de propósito general, y las partes del descriptor de segmento de los registros de segmento.

7. Empieza a ejecutar la nueva tarea.

El estado de la tarea en curso siempre es guardado cuando la conmutación de tarea ocurre satisfactoriamente. Si la tarea se reanuda, la ejecución empieza con la instrucción apuntada por el valor EIP guardado, y los registros se restauran a los valores que tenían antes de que se suspendiera.

En la conmutación de tareas, el nivel de privilegio de la nueva tarea no hereda su nivel de privilegio de la tarea suspendida. La nueva tarea empieza ejecutándose en el nivel de privilegio especifico en el CPL del registro CS que está cargado desde el TSS. El software no necesita realizar verificaciones de privilegios explícitos en un conmutación de tarea debido a que las leyes de privilegio controlan el acceso al TSS.

La duración de una conmutación de tarea es de pocos microsegundos, y, en el caso de ser originada mediante una instrucción CALL, se guarda en el TSS nuevo el TR viejo (TR PREVIO en la figura 13.1) para regresar a la tarea peticionaria de la conmutación, con la instrucción de retorno.

La localización en memoria de los recursos destinados a cada tarea se consigue con el cambio automático del contenido de los registros LDTR y CR3, que apuntan a la tabla de descriptores locales y a la base del Directorio de las Tablas de Páginas.

13.5- PUERTAS DE TAREA.

La conmutación de tarea mediante la ejecución de instrucciones JMP o CALL en un segmento de código de la tarea vieja, exigen que el nivel de privilegio del mismo sea el máximo, es decir, el 0 para igualarlo al del TSS.

Para poder acceder a un TSS desde un segmento de código con menor nivel de privilegio, se usan las puertas de tarea, que son un tipo especial de descriptor del sistema ubicados en la GDT, en la LDT o en la IDT.

La puerta de tarea realiza una indirección sobre el TSS, comportándose de forma similar a la puerta de llamada (figura 13.6).

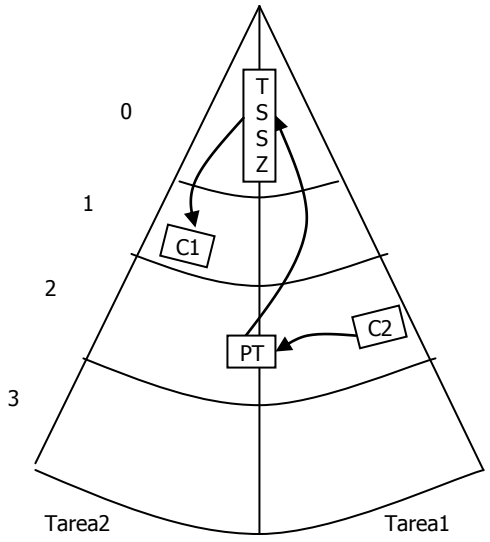


Figura 13.5. Mediante la puerta de tarea (PT) con PL = 2, desde C2 de la tarea 1 se puede realizar la conmutación a la tarea 2.

La estructura de un descriptor correspondiente a una puerta de tarea se ofrece en la figura 13.7 y sólo tiene dos campos útiles:

1. **El selector del TSS de la tarea,** que se carga en el TR del Pentium.

2. **Los atributos**, con los siguientes bits válidos: el bit P de presencia, el campo de dos bits DPL que indica el PL en el que se ha situado la puerta, y el campo TIPO, que identifica el descriptor como puerta de tarea y que tiene el código 5 en hexadecimal.

3. **El DPL (Nivel de privilegio del Descriptor)** es el encargado de producir la conmutación de tarea, puesto que para acceder a la puerta de tarea es preciso poseer un nivel de privilegio igual o mayor.

DESCRIPTOR DE LA PUERTA DE TAREA.

31 — 16	15	8	7 — 0
RESERVADO	**P**	**DPL** (2BITS) / **TIPO 00101**	**RESERVADO**
SELECTTOR DEL TSS	**RESERVADO**		

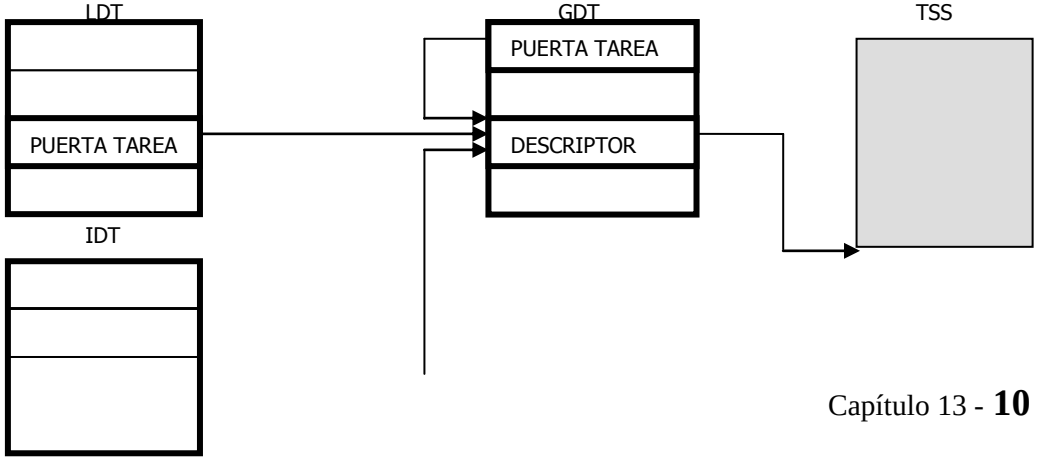
Figura 13.6. En un descriptor de una puerta de tarea, sólo están definidos el campo del selector del TSS y los atributos.

Se puede acceder a una tarea a través de un descriptor de puerta de tarea o a través de un TSS. Ambas estructuras se proporcionan para satisfacer las siguientes necesidades:

1. La necesidad de una tarea de tener sólo un flag BUSY. Como el flag BUSY se guarda para una tarea en el descriptor TSS, cada tarea debe tener sólo un descriptor TSS. Sin embargo, puede haber varias puertas de tareas refiriéndose al mismo descriptor TSS.

2. La necesidad de proporcionar el acceso selectivo a las tareas. Las puertas de tarea llenan esta necesidad, porque éstas pueden residir en una LDT y pueden tener un DPL que es diferente al DPL del descriptor de TSS. Un programa o proceso que no tiene el suficiente privilegio para acceder al descriptor TSS para una tarea en el GDT (qué normalmente tiene un DPL de 0) puede permitirse el acceso a la tarea a través de la puerta de tarea con un DPL más alto. Las puertas de tarea dan al sistema operativo el nivel mayor para limitar el acceso a las tareas específicas.

3. La necesidad de que una interrupción o excepción sea manejada por una tarea independiente. Las puertas de tarea también pueden residir en el IDT que permite manejar las interrupciones y excepciones por las tareas del "handler". Cuando una interrupción o el vector de excepciones apunta a una puerta de tarea, la CPU cambia a la tarea especificada a través de la citada puerta de tarea. En la figura 13.8 se ilustra cómo una puerta de tarea en un LDT, una puerta de tarea en un GDT, y una puerta de tarea en un IDT, pueden apuntar a la misma tarea.



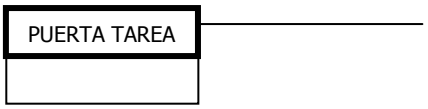


Figura 13.8. Puertas de tarea referenciando a una misma tarea

Cuando, desde el programa en curso, se ejecuta una instrucción JMP o CALL sobre una puerta de tarea en principio se comprueban las reglas de acceso comparando el CPL del programa en ejecución con el DPL de la puerta. Si la regla de acceso relativa a los PL se cumple, se salva el contexto de la CPU en el TSS de la tarea actual, referenciado por el contenido de la TR. Después, el descriptor de la puerta de tarea proporciona el selector del nuevo TSS que se carga en TR, procediendo a cargar la información almacenada en el TSS nuevo en la CPU, es decir, a configurar un nuevo contexto. A partir de este momento, el proceso inicia la nueva tarea con la instrucción direccionada por la LDT y CS:EIP, que ha suministrado el TSS nuevo (figura 13.9).

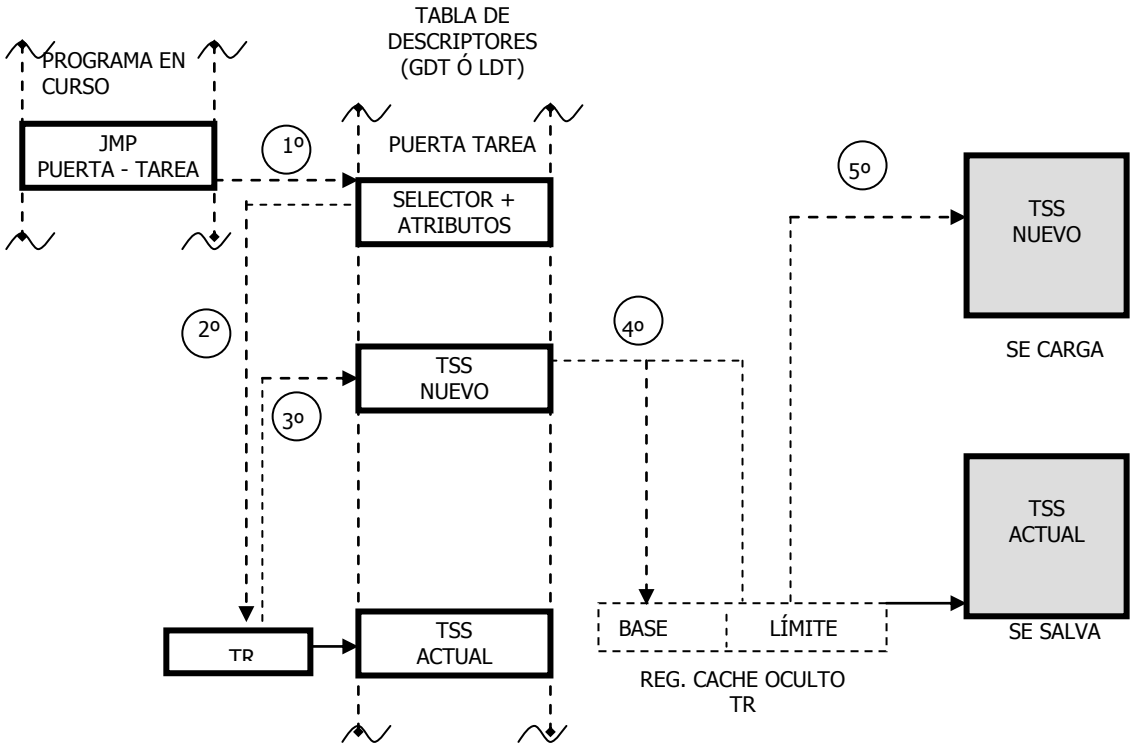


Figura 13.9. Expresión gráfica del mecanismo de conmutación de tarea a través de una puerta de tarea.

Al poder residir los descriptores de las puertas de tarea, indistintamente, en la GDT o en la LDT, es posible solicitar la conmutación de tarea desde cualquier punto del sistema.

En la conmutación de tarea el mecanismo de protección comprueba el cumplimiento de diversas reglas:

1. El nivel de privilegio de la puerta de tarea ha de ser igual o menor que el EPL, que es el máximo entre el CPL y el RPL del selector de la puerta.

2. El descriptor del TSS nuevo debe estar presente (P) en la memoria y con el límite correcto.

3. Cuando se carga el selector del TSS nuevo en el TR, hay que marcarlo como ocupado.

4. Para indicar que se ha producido una conmutación de tarea, se pone a 1 el bit TS de la Palabra de Estado.

Para soportar adecuadamente los anidamientos entre tareas, realizados mediante instrucciones CALL, el Pentium maneja el bit B (ocupado o BUSY) existente en el descriptor del TSS.

En general, en una conmutación de tareas el Pentium pone a 1 el bit B del descriptor del TSS de la tarea nueva, que comienza a procesarse, y pone a 0 el bit B del descriptor del TSS viejo. Esto último sólo sucede si la conmutación se ha originado como consecuencia de algún hecho que no sea la ejecución de una instrucción CALL, pues, en tal caso, permanece a 1 el bit B de la tarea anterior.

De esta forma, en los anidamientos de tareas, al aparecer la instrucción IRET de retorno, la CPU utiliza el valor del TR previo guardado en el TSS activo, para conmutar con la tarea que antes la enlazó o anidó.

El empleo de las puertas de tarea se presenta muy eficaz cuando se desea que una determinada subrutina proporcione servicio a un conjunto de varias tareas. Ejecutando dicha subrutina como una tarea independiente, se asegura que el contexto de la tarea en curso no sea corrompido al manejar la tarea común, pues en la conmutación se salva el estado de la tarea en curso (figura 13.10).

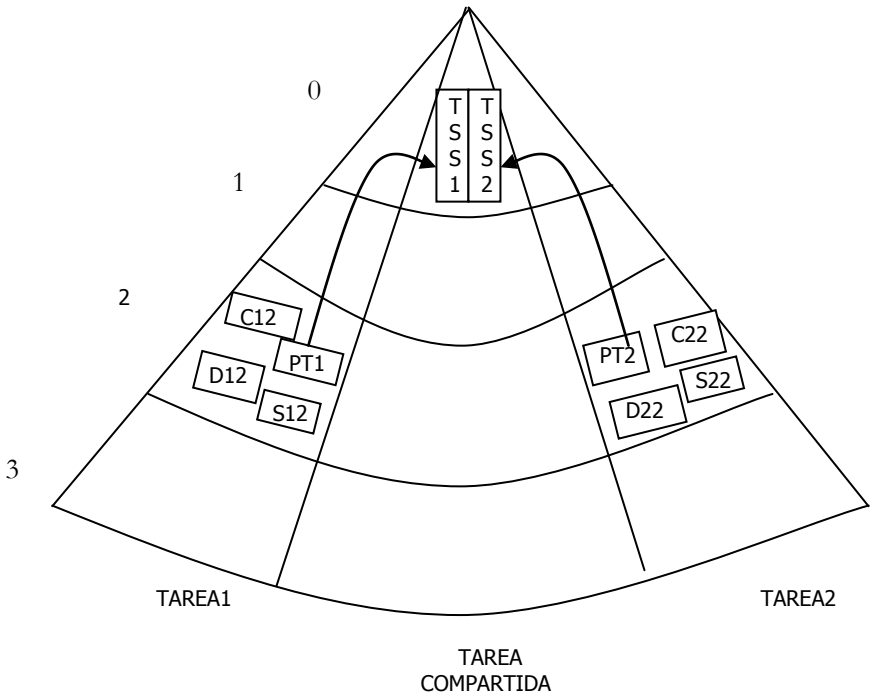


Figura 13.10. Las puertas de tarea permiten crear un entorno seguro en ambientes multitarea en los que existe una tarea compartida.

13.6- MAPA DE BITS DE PERMISO DE E/S.

El segmento TSS tiene una longitud mínima de 67H bytes y en él se guarda el contexto fundamental de la CPU. Los cuatro bytes situados en su cima, cuyas direcciones en hexadecimal

serán las 64, 65, 66 y 67, en estas posiciones se encuentra, ocupando los bytes 66 y 67, el campo llamado "base del mapa de permiso de E/S".

Los 16 bits del campo base del mapa de permiso de E/S, conforman el desplazamiento que hay que añadir a la posición que ocupa este campo, o sea, 67 H, para apuntar el comienzo del "mapa de bits de permiso de E/S". Dicho mapa ocupa desde la posición anteriormente citada hasta el límite del segmento TSS.

El mapa de bits de permiso de E/S está compuesto por una serie de bits 0 y 1, cada uno de los cuales se corresponde con la dirección de una puerta de ocho bits del mapa de E/S. Si el bit del mapa es 0, la puerta correspondiente del mapa de E/S es accesible, pero si el bit es 1, la puerta no es accesible y se produce una excepción de violación de la protección en el caso de que intente acceder a ella.

En el caso de que una instrucción de E/S afecte a un operando de tamaño mayor que el byte es decir, se quiera realizar una operación de E/S con una información de más de ocho bits (una puerta de E/S), todos los bits que referencien las puertas a las que afecta dicha instrucción deben estar a cero.