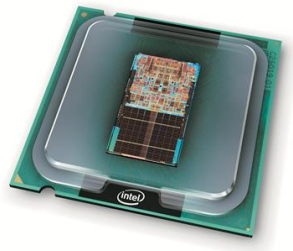


Procesadores IA-32 e Intel® 64
Tareas

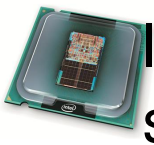
Alejandro Furfaro

Junio de 2012

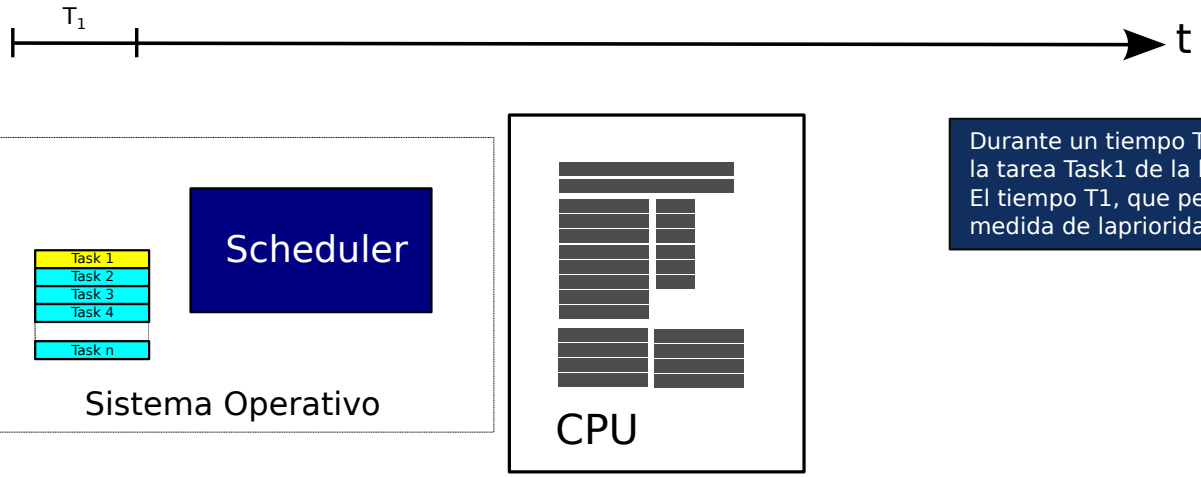
Temario

¿Quien o quienes conmutan tareas?

- Es una tarea compartida entre el Procesador y el Sistema Operativo.
- No funciona simultáneamente, sino en forma serializada a gran velocidad
- Nuestros sentidos no captan la intermitencia de cada tarea, creándose una sensación de simultaneidad.
- Para ello el sistema operativo tiene
  - un módulo de software llamado ***scheduler***
  - Una lista de tareas a ejecutar
  - Un intervalo de tiempo llamado time frame dividido en intervalos mas pequeños de modo de asignarle a cada tarea un porcentaje del time frame.
- Cada tarea tiene así unos milisegundos para progresar, expirados los cuales suspende una tarea y despacha para su ejecución la siguiente de la lista.

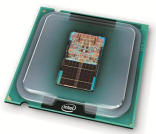


¿Como lo hacen?

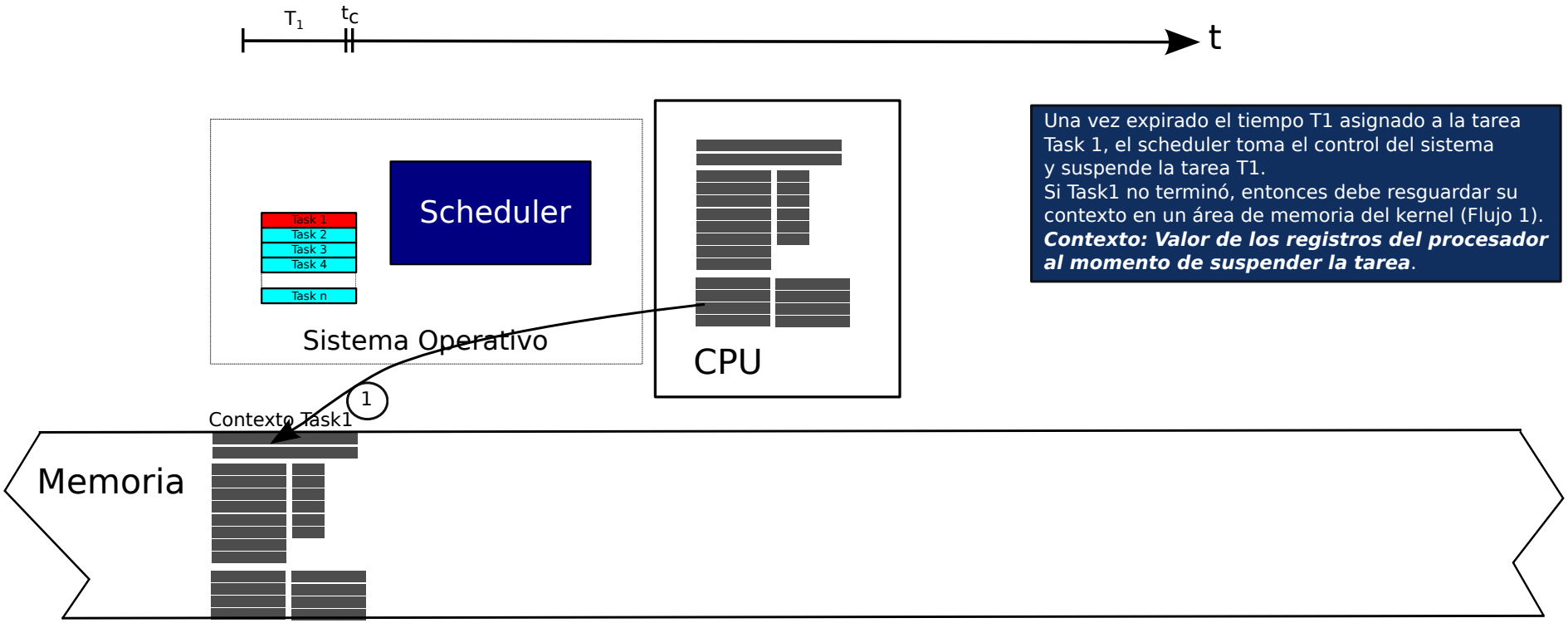


Durante un tiempo T1, el procesador ejecutará la tarea Task1 de la lista que maneja el scheduler. El tiempo T1, que permanezca ejecutandola es medida de laprioridad de esta tarea.

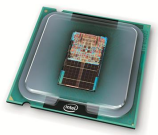
Memoria



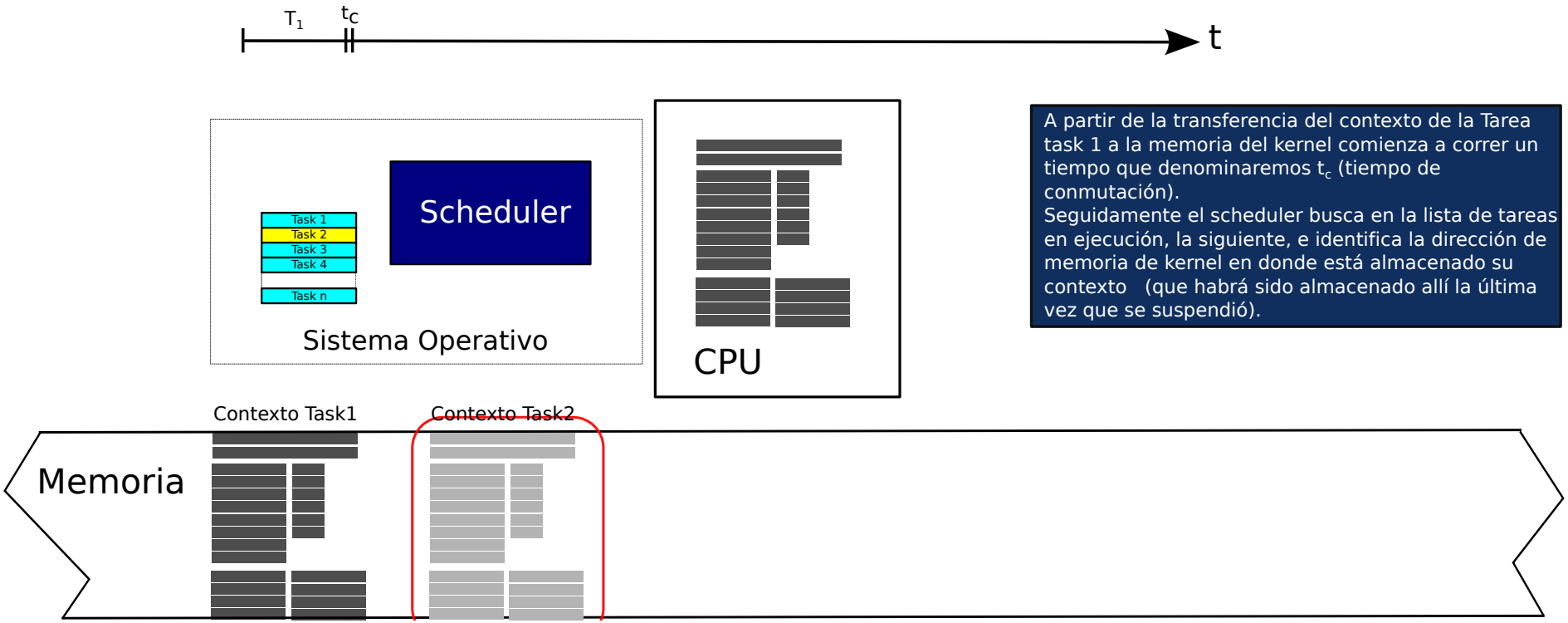
¿Como lo hacen?



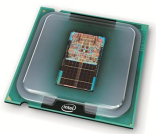
Una vez expirado el tiempo T1 asignado a la tarea Task 1, el scheduler toma el control del sistema y suspende la tarea T1.
Si Task1 no terminó, entonces debe resguardar su contexto en un área de memoria del kernel (Flujo 1).
Contexto: Valor de los registros del procesador al momento de suspender la tarea.



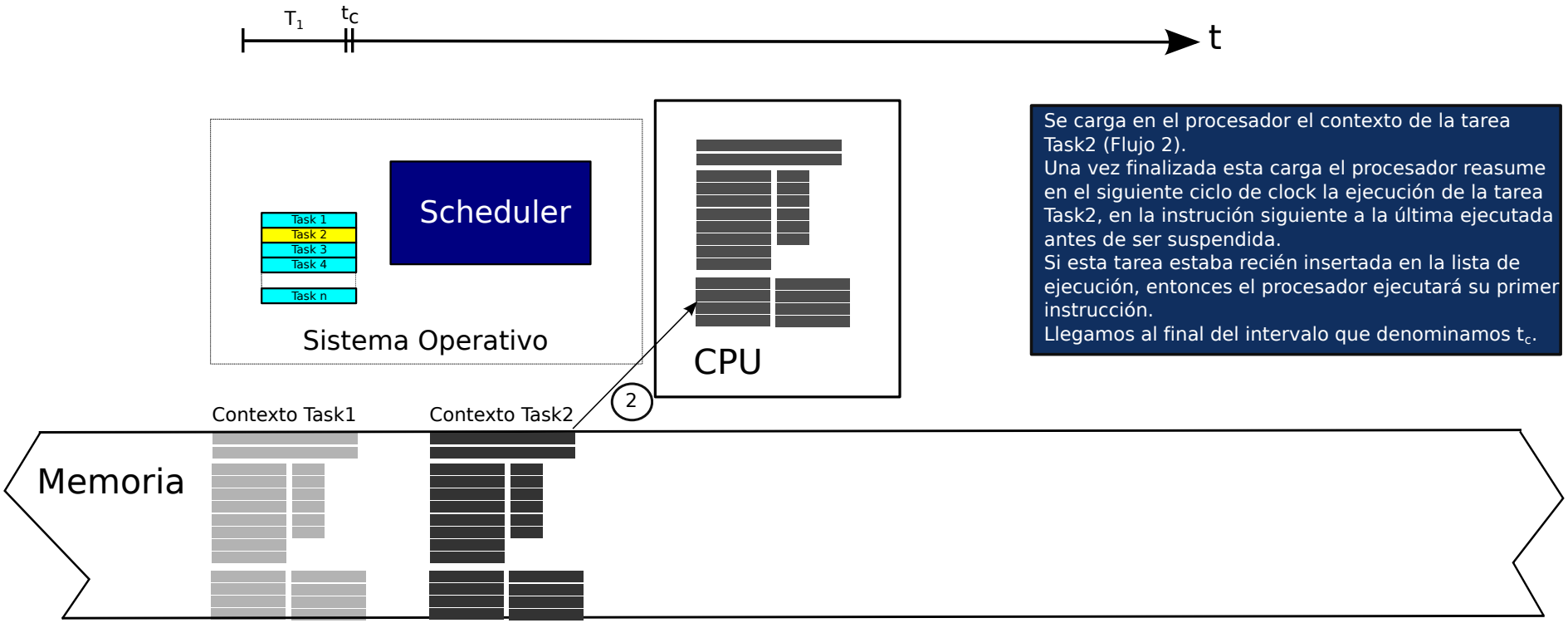
¿Como lo hacen?



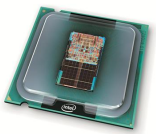
A partir de la transferencia del contexto de la Tarea task 1 a la memoria del kernel comienza a correr un tiempo que denominaremos t_c (tiempo de conmutación).
Seguidamente el scheduler busca en la lista de tareas en ejecución, la siguiente, e identifica la dirección de memoria de kernel en donde está almacenado su contexto (que habrá sido almacenado allí la última vez que se suspendió).



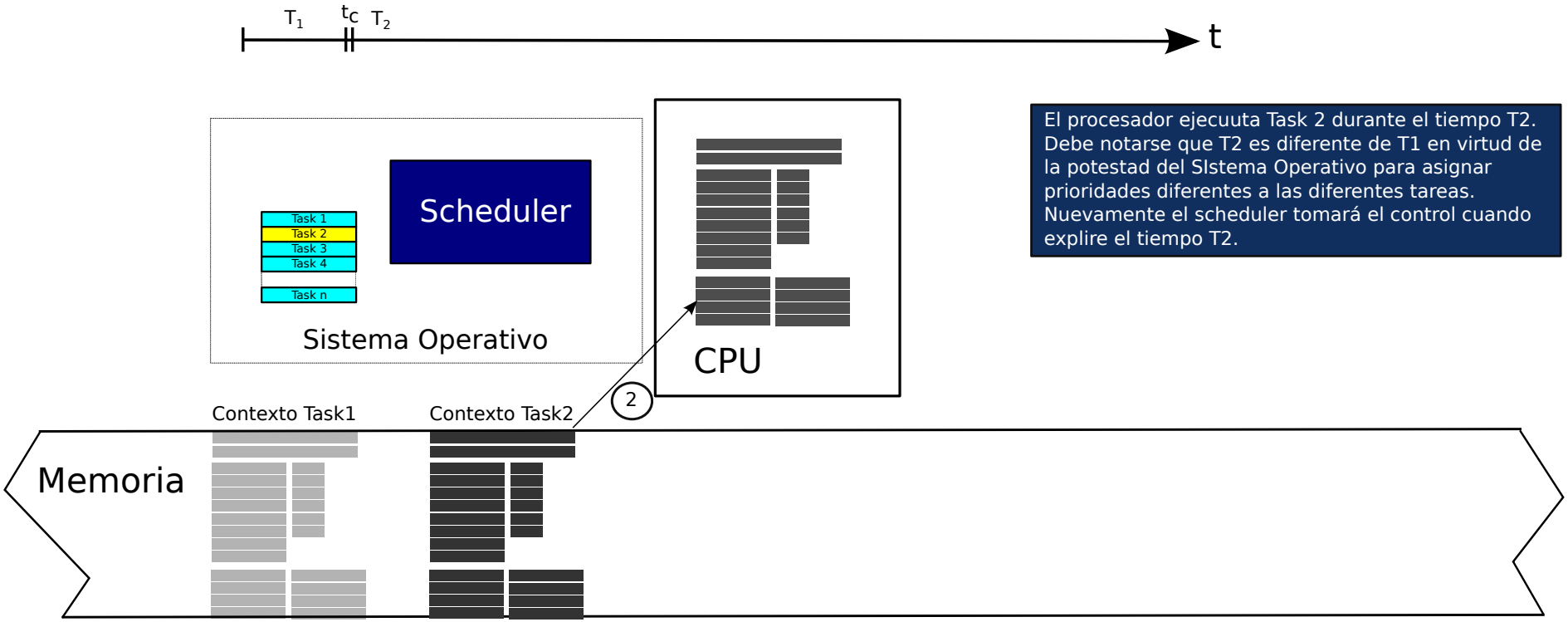
¿Como lo hacen?



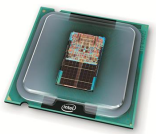
Se carga en el procesador el contexto de la tarea Task2 (Flujo 2).
Una vez finalizada esta carga el procesador reasume en el siguiente ciclo de clock la ejecución de la tarea Task2, en la instrución siguiente a la última ejecutada antes de ser suspendida.
Si esta tarea estaba recién insertada en la lista de ejecución, entonces el procesador ejecutará su primer instrucción.
Llegamos al final del intervalo que denominamos t_c.



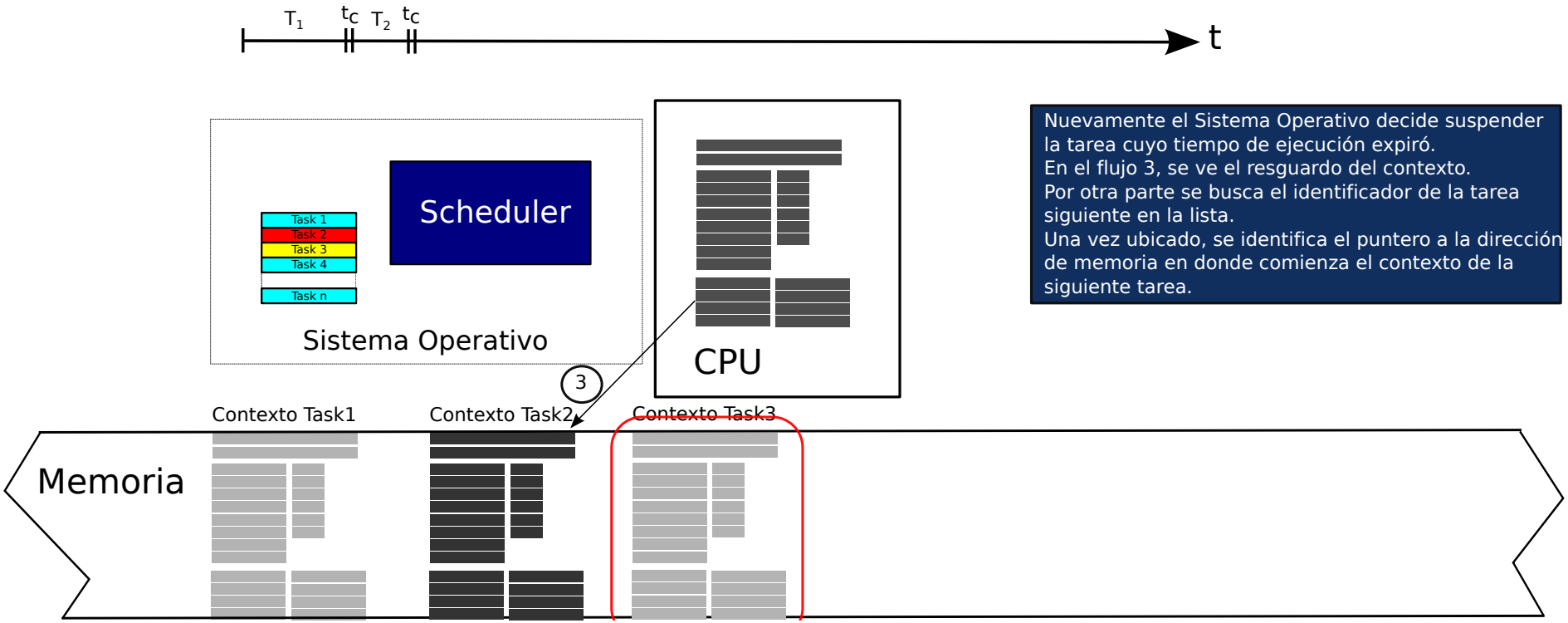
¿Como lo hacen?



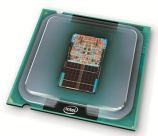
El procesador ejecuuta Task 2 durante el tiempo T2. Debe notarse que T2 es diferente de T1 en virtud de la potestad del SIstema Operativo para asignar prioridades diferentes a las diferentes tareas. Nuevamente el scheduler tomará el control cuando explire el tiempo T2.



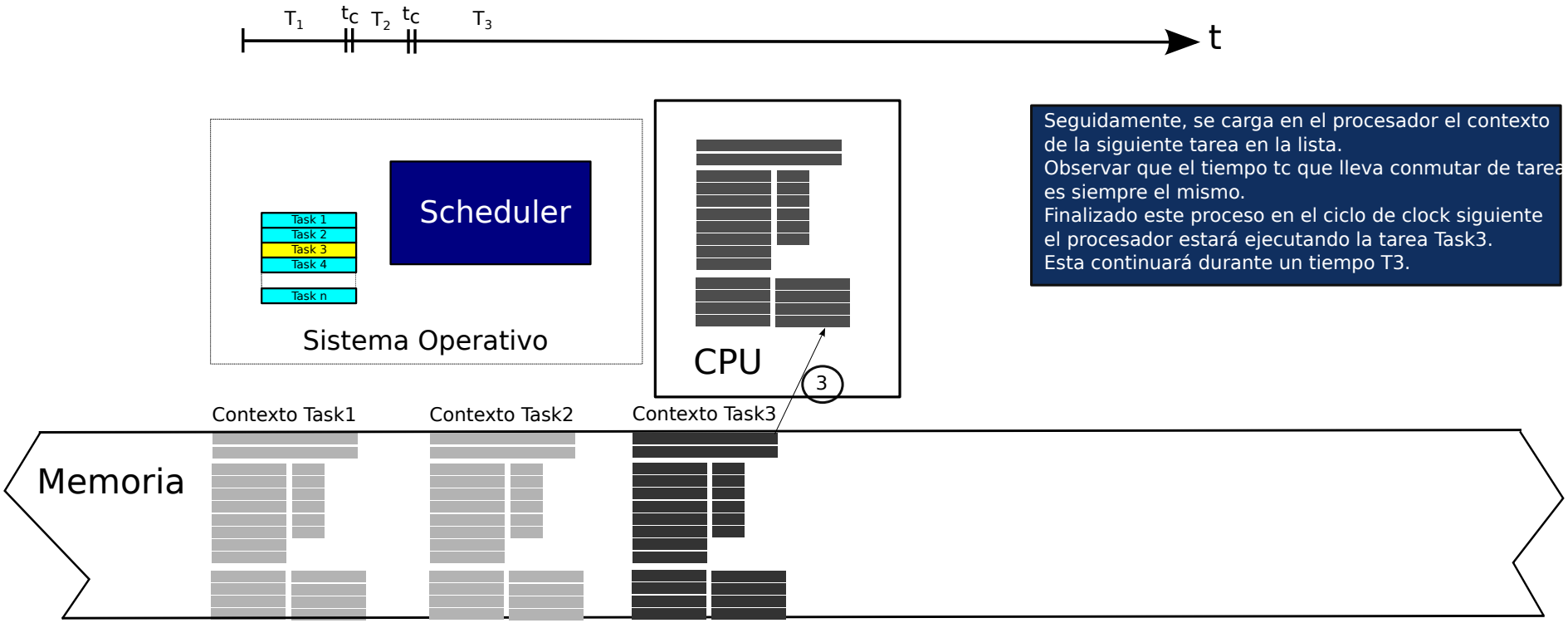
¿Como lo hacen?



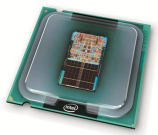
Nuevamente el Sistema Operativo decide suspender la tarea cuyo tiempo de ejecución expiró.
En el flujo 3, se ve el resguardo del contexto.
Por otra parte se busca el identificador de la tarea siguiente en la lista.
Una vez ubicado, se identifica el puntero a la dirección de memoria en donde comienza el contexto de la siguiente tarea.



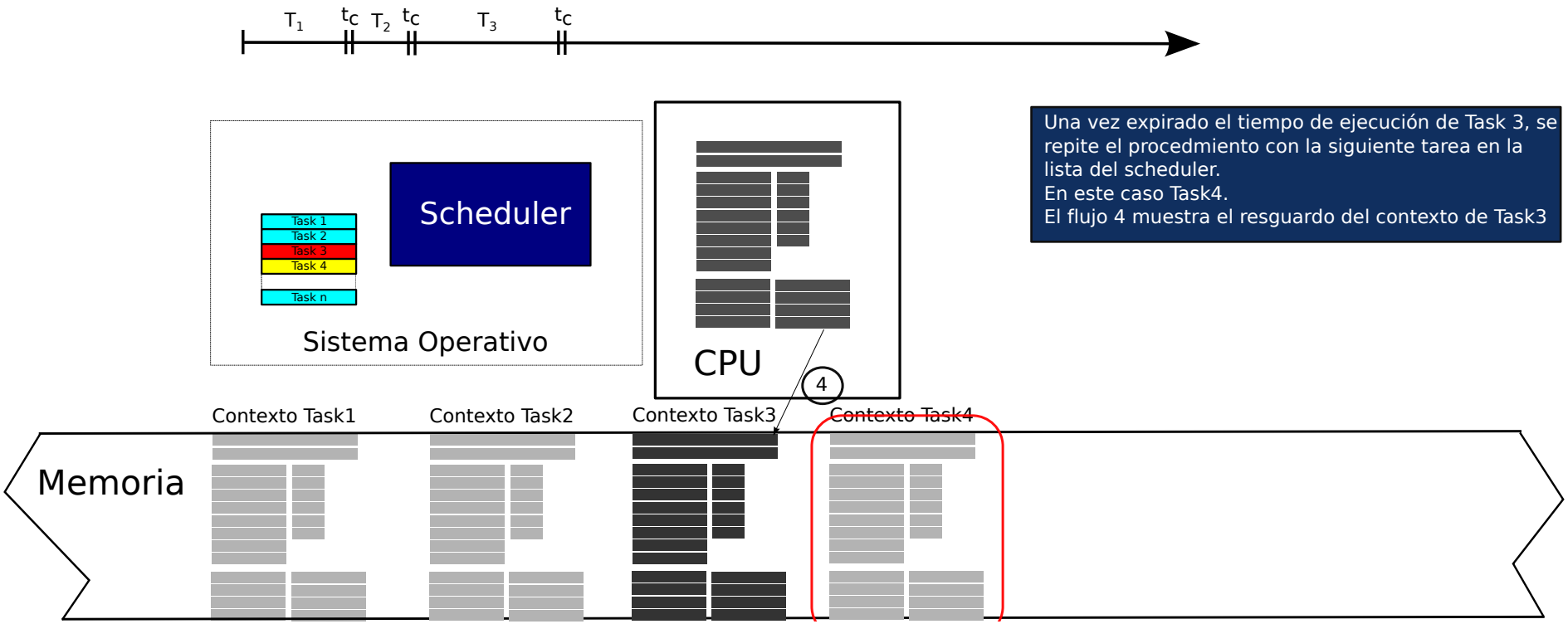
¿Como lo hacen?



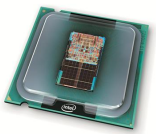
Seguidamente, se carga en el procesador el contexto de la siguiente tarea en la lista.
Observar que el tiempo tc que lleva conmutar de tarea es siempre el mismo.
Finalizado este proceso en el ciclo de clock siguiente el procesador estará ejecutando la tarea Task3.
Esta continuará durante un tiempo T3.



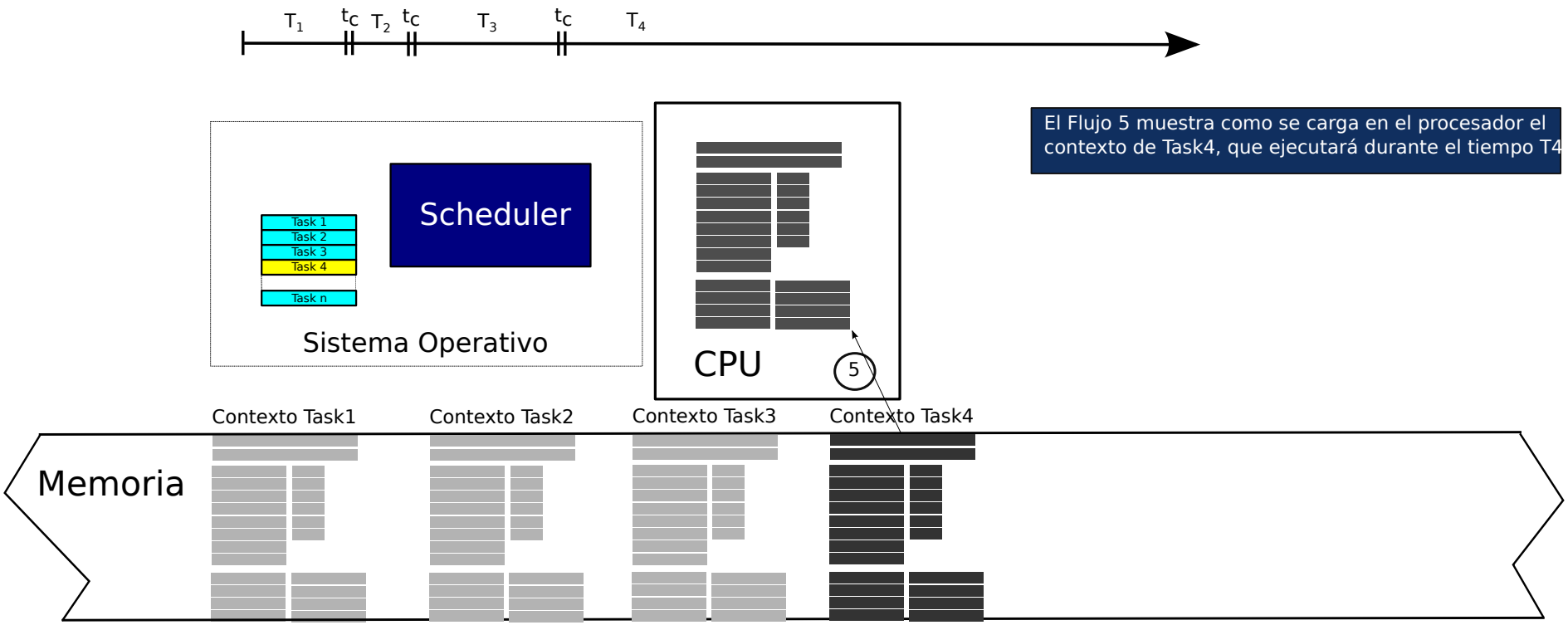
¿Como lo hacen?

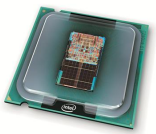


Una vez expirado el tiempo de ejecución de Task 3, se repite el procedmiento con la siguiente tarea en la lista del scheduler.
En este caso Task4.
El flujo 4 muestra el resguardo del contexto de Task3

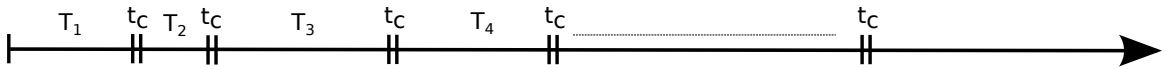


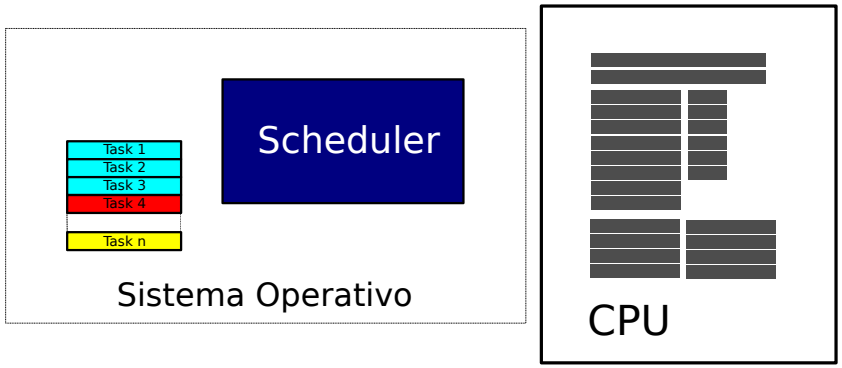
¿Como lo hacen?



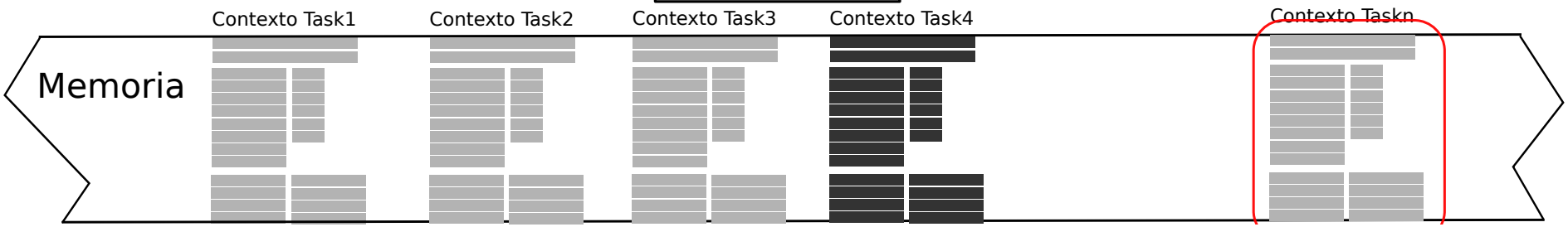


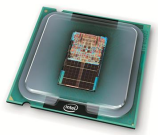
¿Como lo hacen?





Finalmente llega el turno de la última tarea en la lista. Una vez completada esta el scheduler volverá al inicio de la lista y reasumirá ta tarea Task1.





Definiciones

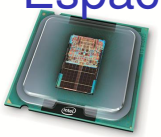
Tarea: es una unidad de trabajo que un procesador puede despachar, ejecutar, y detener a voluntad, bajo la forma de:

- La instancia de un programa (o, expresado en términos del Sistema Operativo, proceso).
- Un handler de interrupción.
- Un servicio del kernel (Núcleo del Sistema Operativo).

Espacio de ejecución: Es el conjunto de segmentos de código, datos, y pila que componen la tarea. En un sistema operativo que utilice los mecanismos de protección del procesador se requiere un segmento de pila por cada nivel de privilegio.

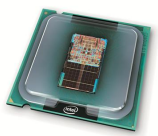
Contexto de ejecución: Es el conjunto de valores de los registros internos del procesador en cada momento. Para poder suspender la ejecución de una tarea y poder reasumirla posteriormente, es necesario almacenar este contexto en el momento de la suspensión.

Espacio de Contexto de ejecución: Se compone de un segmento de memoria en el que el kernel del S.O. almacenará el contexto completo de ejecución del procesador. Se lo denomina Task State Segment (TSS)



La arquitectura se basa en 5 elementos

1. Segmento de estado de tarea (TSS).
2. Descriptor de TSS
3. Descriptor de Puerta de Tarea
4. Registro de tarea
5. Flag NT (bit 14 de EFLAGS)



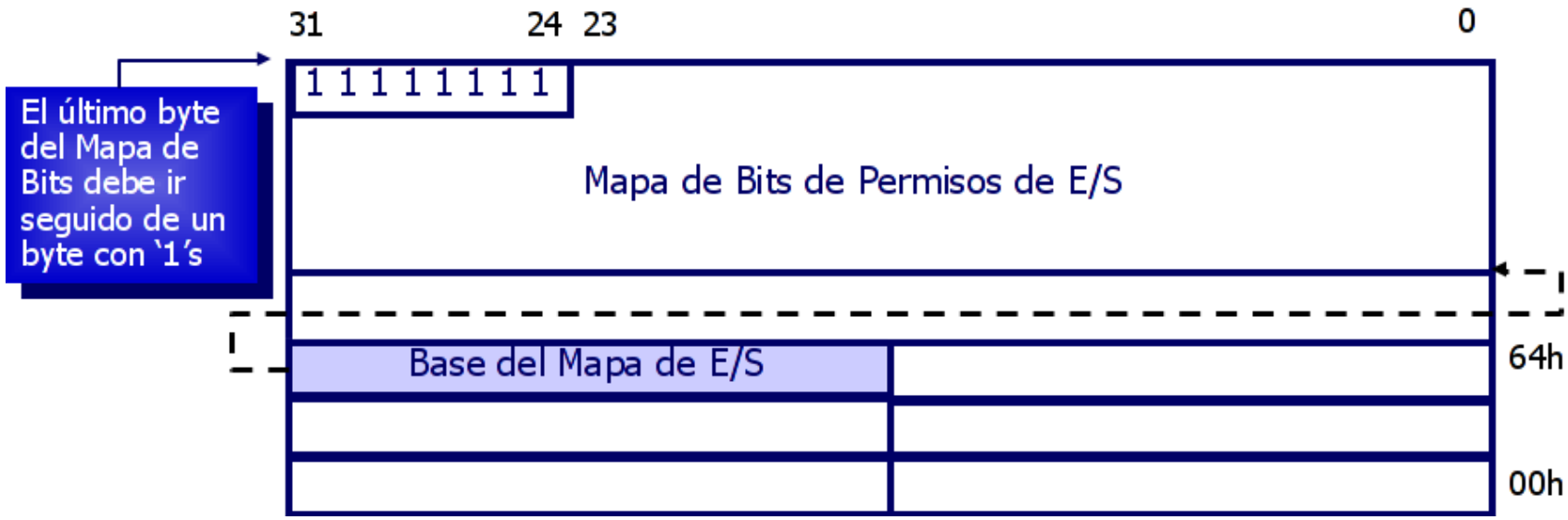
El TSS implementa el Espacio de Contexto

31	15	0	
I/O Map Base Address	Reserved	T	100
Reserved	LDT Segment Selector		96
Reserved	GS		92
Reserved	FS		88
Reserved	DS		84
Reserved	SS		80
Reserved	CS		76
Reserved	ES		72
EDI			68
ESI			64
EBP			60
ESP			56
EBX			52
EDX			48
ECX			44
EAX			40
EFLAGS			36
EIP			32
CR3 (PDBR)			28
Reserved	SS2		24
ESP2			20
Reserved	SS1		16
ESP1			12
Reserved	SS0		8
ESP0			4
Reserved	Previous Task Link		0

Reserved bits. Set to 0.

1. Es el lugar de memoria previsto en los procesadores IA-32 como espacio de contexto de cada tarea.
2. El tamaño mínimo de este segmento es 67h.
3. Se guardan los valores de SS y ESP para los stacks de nivel 2, 1, y 0. El del nivel 3 eventualmente estará en los registros SS:ESP.
4. El Flag T genera una excepción de Debug cada vez que se conmuta de tarea (Pentium Pro en adelante), si está en ‘1’.
5. I/O Map Base Address: Offset de 16 bits desde el inicio del TSS hasta el inicio del Mapa de permisos de E/S

Mapa de Bits de E/S

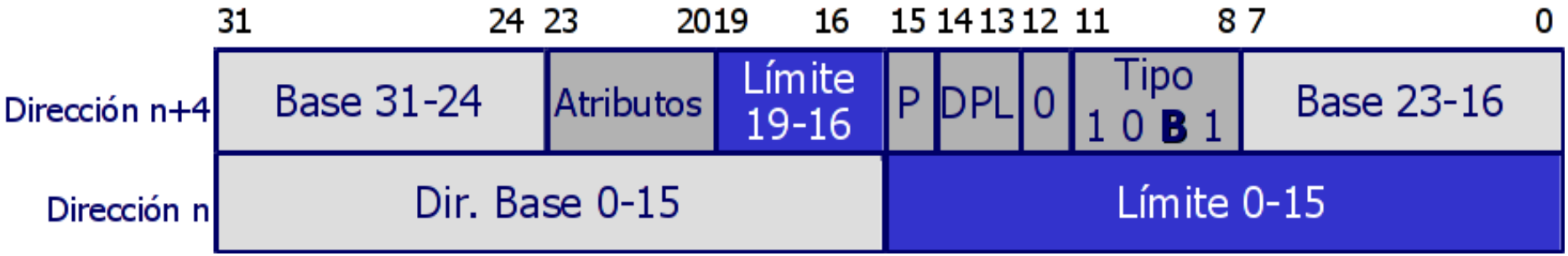


- En Modo Protegido, por default una tarea que ejecuta con **CPL=11** no puede ejecutar instrucciones de acceso a E/S, es decir IN, OUT, INS, y OUTS.
- El procesador utiliza el par de bits **IOPL** del registro **EFLAGS**, para modificar este comportamiento default, de manera selectiva para cada tarea.
- Para una determinada tarea con CPL=11, si desde el S.O. se pone el campo **IOPL** de **EFLAGS** en 11, se habilita el acceso a las direcciones de E/S cuyos bit correspondientes estén seteados en el Bit Map de permisos de E/S.

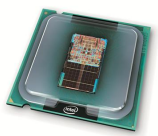
Así se habilita el acceso a determinados ports para determinadas tareas.

- En este caso el TSS mide mas de 104 bytes (su límite será mayor que 67h).

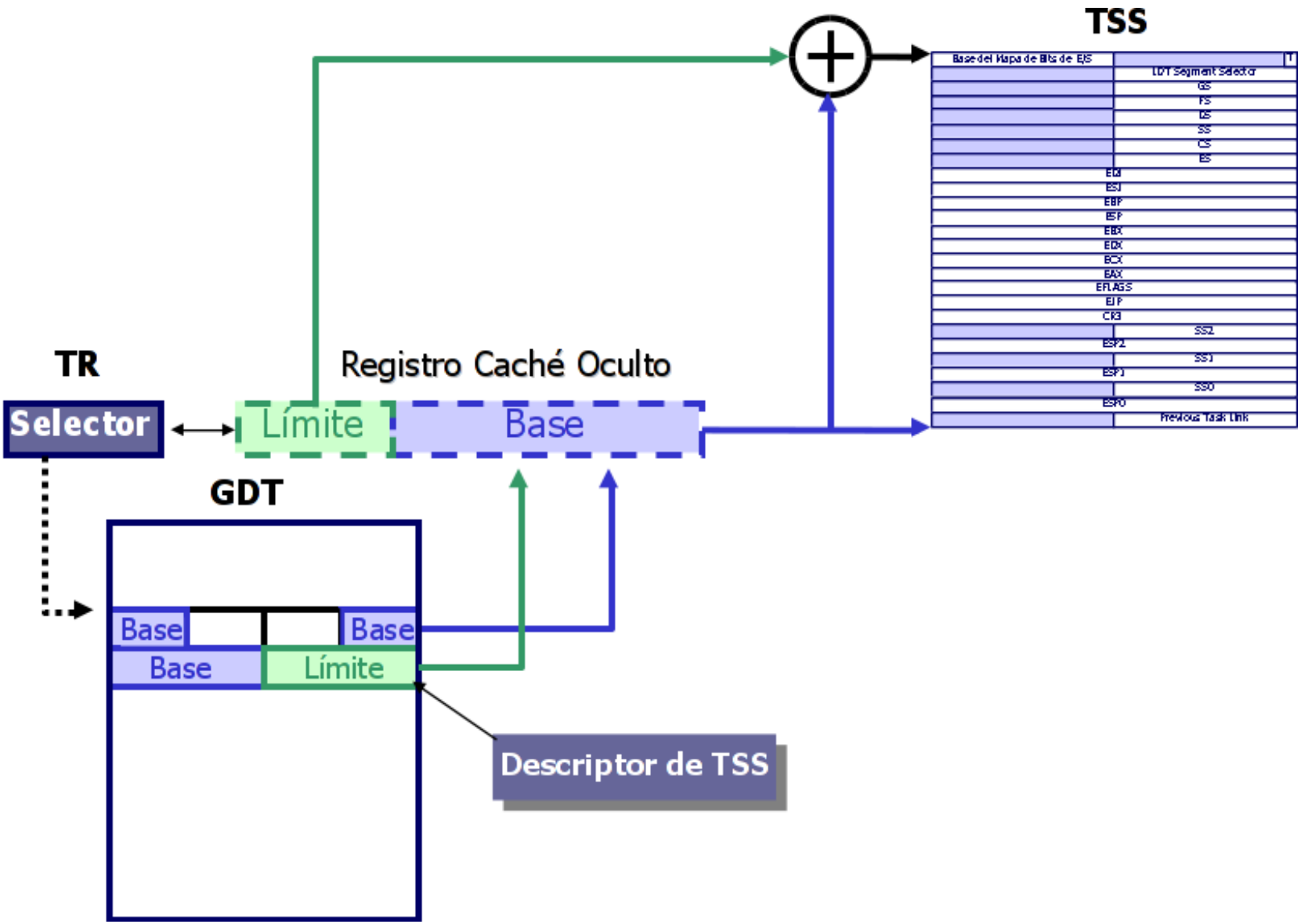
Cada TSS necesita un descriptor



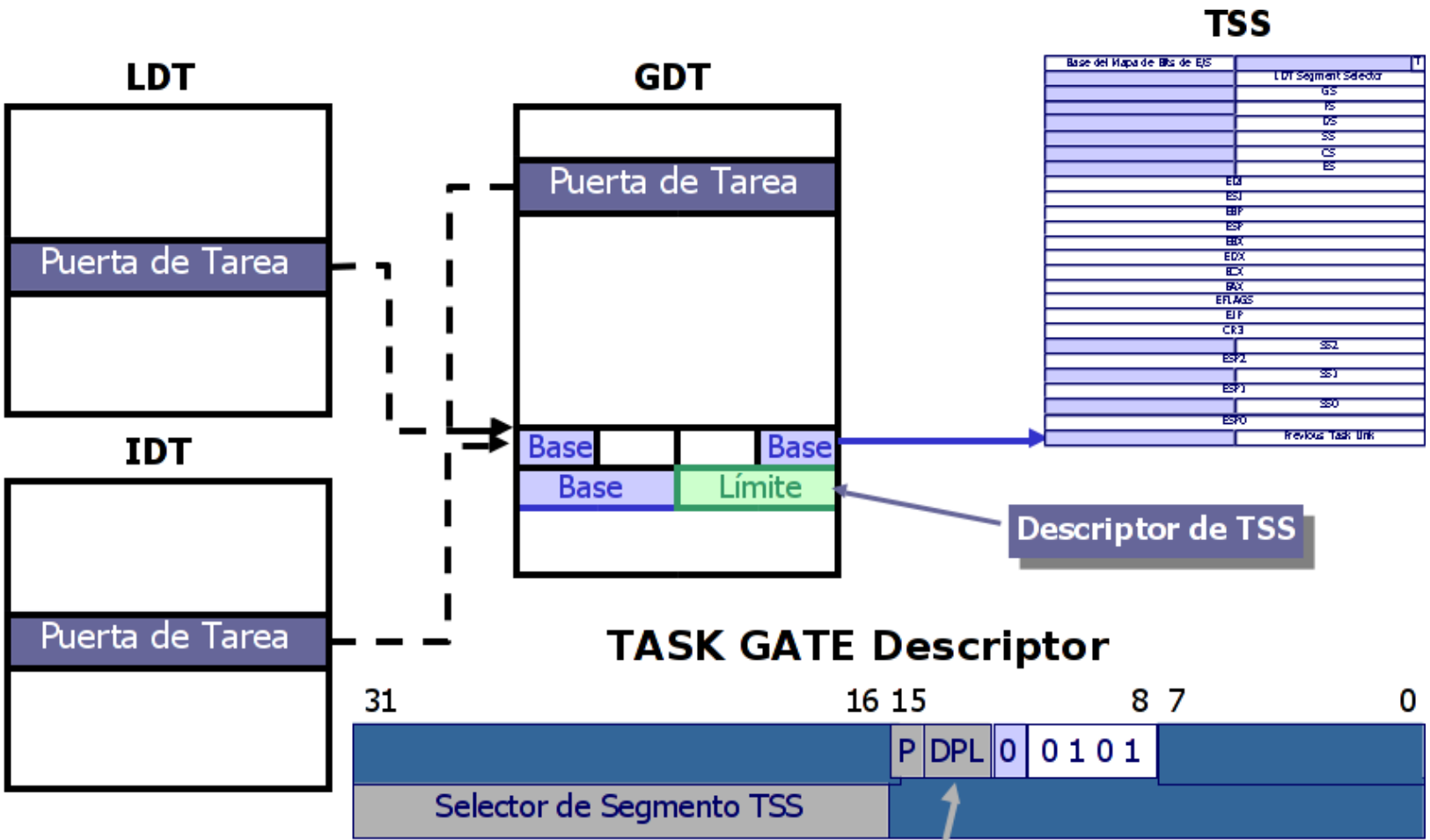
- El Bit **B** (Busy) sirve para evitar recursividad en el anidamiento de tareas. Nos referiremos a él con mas detale cuando analicemos el anidamiento de tareas.
- El Límite debe ser mayor o igual a 67h (mínimo tamaño del segmento es 0x68, o 103_{10}. De otro modo se genera una excepción TSS inválido, tipo 0x0A.

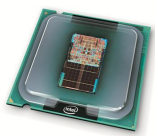


Cada TSS necesita un descriptor



Task Gate: otra forma de acceder a una tarea

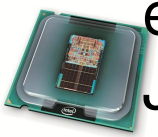




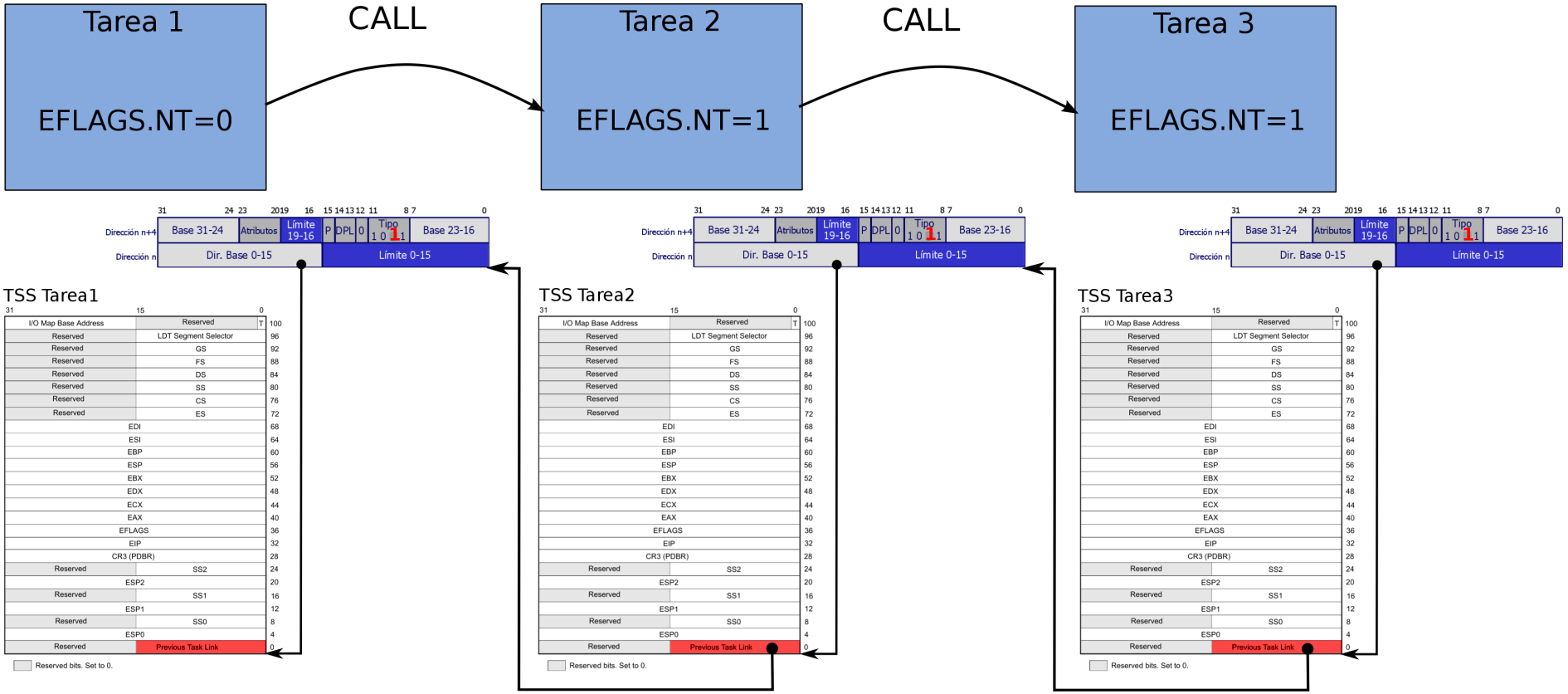
Si DPL=11 en un task gate, aún desde una tarea con CPL 11, puede efectuarse una conmutación de tareas. Este es otro mecanismo para acceder al kernel desde una aplicación.

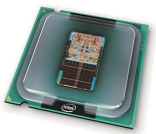
¿Como se cambia a una tarea?

- El procesador puede despachar una tarea de las siguientes formas posibles:
  1. Por medio de una instrucción CALL
  2. Por medio de una instrucción JMP
  3. Mediante una llamada implícita del procesador al handler de una interrupción manejado por una tarea.
  4. Mediante una llamada implícita del procesador al handler de una excepción manejado por una tarea.
  5. Mediante la ejecución de la instrucción IRET en una tarea cuando el flag NT (bit 14 del registro EFLAGS) es “1” para la tarea actual.
- En cualquier caso se requiere poder identificar a la tarea.
- Se necesita un selector en la GDT que apunte a una puerta de tarea o a un Task State Segment (TSS). Este selector debe estar en la correspondiente posición dentro de la instrucción CALL o JMP.



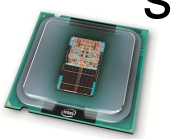
Funcionamiento de tareas anidadas





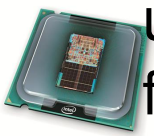
Funcionamiento de tareas anidadas

- Cuando se conmuta a una tarea mediante un CALL, una interrupción, o una excepción, el procesador copia el TR de la tarea actual en el campo “Previous Task Link” del TSS de la nueva tarea, y luego de completar el cambio de contexto, setea el bit NT del registro EFLAGS (bit 14).
- De este modo si la nueva tarea ejecuta en algún punto la instrucción IRET y el bit NT es ‘1’, el procesador conmuta a la tarea anterior ya que tiene el selector de TSS de la tarea previa almacenado en el campo Previous Task Link del TSS de la tarea en ejecución.
- En cambio si la conmutación de tarea se efectúe con un JMP no se afecta el flag NT ni se completa el campo “Previous Task Link.



Funcionamiento de tareas anidadas

- El procesador utiliza el Bit Busy de un descriptor de TSS para prevenir la reentrancia en una tarea (esto ocasionaría la pérdida de los datos del contexto de ejecución).
- Cuando se avanza en anidamiento de tareas mediante un CALL o una interrupción, este bit debe permanecer seteado, en el descriptor de TSS de la tarea previa.
- El procesador generará una Excepción de Protección General #GP, si se intenta despachar mediante un CALL o una Interrupción una tarea en cuyo TSS está seteado el bit Busy. Esto no ocurre si la la tarea en cuestión se despacha con un IRET ya que es de esperar en esta condición que el bit Busy esté seteado.
- Cuando la tarea ejecuta IRET o bien mediante un JMP despacha una nueva tarea, el procesador asume que la tarea actual finalizará y se debe limpiar el bit Busy en su descriptor de TSS



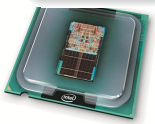
Uno de los cambios mayores...

¿Que cambia en Modo IA-32e?

Se mantienen los conceptos de espacio de tareas, estado de tareas, y se deben construir también las estructuras para almacenar los contextos de cada tarea. Sin embargo, no se mantiene el mecanismo de soporte a la conmutación de tareas propio del Modo Protegido legacy.

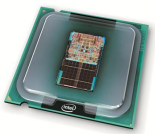
Condiciones que generan una excepción #GP

- Si se transfiere el control a un TSS o a una task gate mediante las instrucciones JMP, CALL, INT, o mediante una interrupción de hardware.
- Si se ejecuta IRET con EFLAGS.NT=1.



¿Que pasa con el TSS?

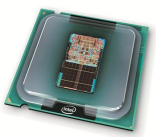
- En Modo 64 bits el procesador mantiene el TSS, pero su función ya no es la de almacenar el contexto de la tarea.
- Su función ahora es mantener
  1. Los valores de RSP para los Niveles de Privilegio 2, 1, y 0, en formato canónico.
  2. La Tabla de Punteros a Stacks de Interrupciones (IST), punteros expresados también en su formato canónico.
  3. El Offset al BitMap de E/S.
- El sistema operativo de 64 debe crear al menos un TSS e inicializar el TR con el selector correspondiente a este segmento.
- Este segmento se utilizará tanto para tareas que ejecuten en el sub-modo 64 bits como en el sub-modo compatibilidad.



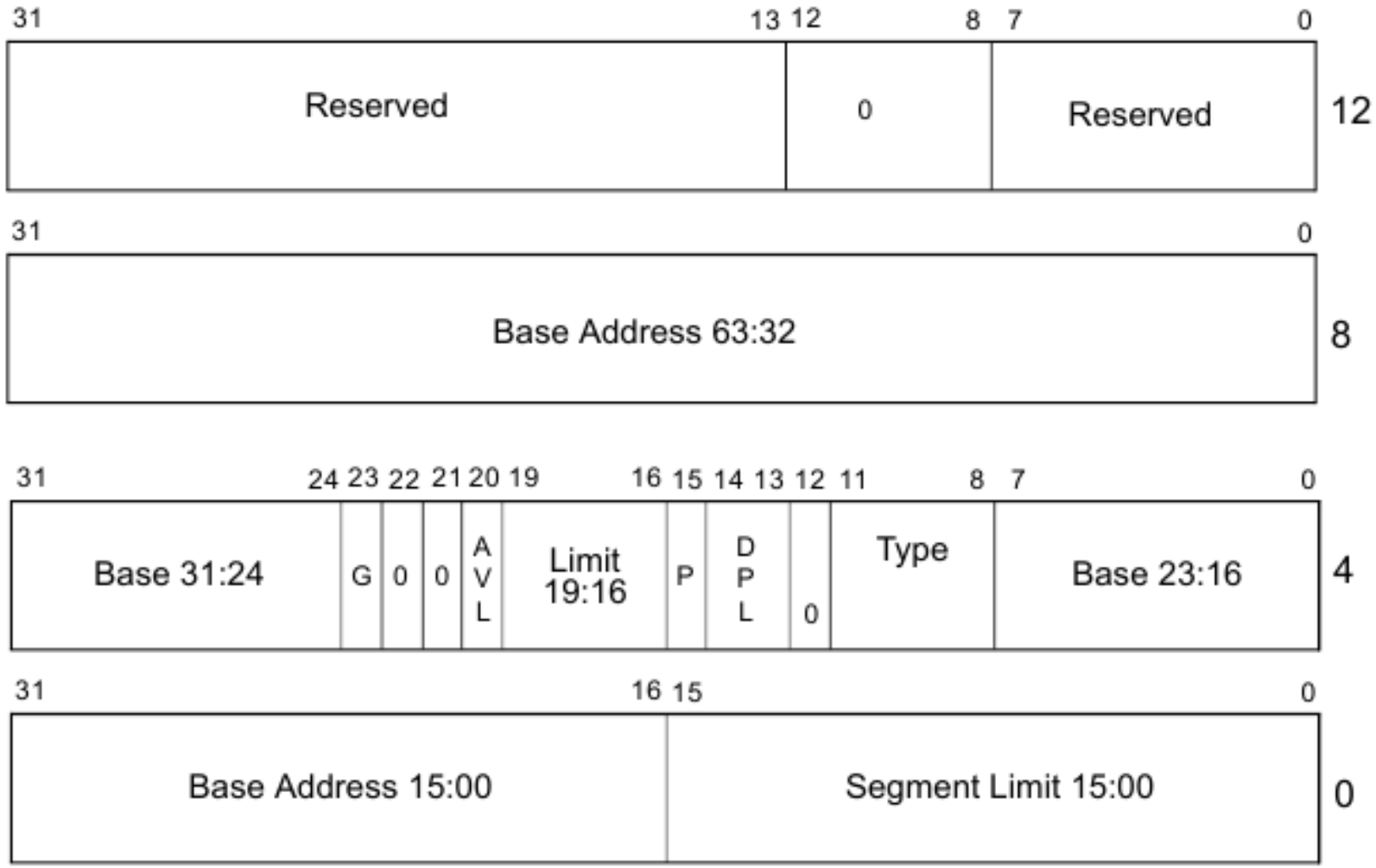
TSS en Modo IA-32e

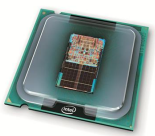
31 — 15	0	
I/O Map Base Address	Reserved	100
Reserved		96
Reserved		92
IST7 (upper 32 bits)		88
IST7 (lower 32 bits)		84
IST6 (upper 32 bits)		80
IST6 (lower 32 bits)		76
IST5 (upper 32 bits)		72
IST5 (lower 32 bits)		68
IST4 (upper 32 bits)		64
IST4 (lower 32 bits)		60
IST3 (upper 32 bits)		56
IST3 (lower 32 bits)		52
IST2 (upper 32 bits)		48
IST2 (lower 32 bits)		44
IST1 (upper 32 bits)		40
IST1 (lower 32 bits)		36
Reserved		32
Reserved		28
RSP2 (upper 32 bits)		24
RSP2 (lower 32 bits)		20
RSP1 (upper 32 bits)		16
RSP1 (lower 32 bits)		12
RSP0 (upper 32 bits)		8
RSP0 (lower 32 bits)		4
Reserved		0

Reserved bits. Set to 0.



Descriptor de TSS en Modo IA-32e





Resumiendo...

En modo 64 Bits la conmutación de tareas no está soportada por el hardware, y por lo tanto debe hacerse por software quedando a cargo del programador de Sistemas....ouch!