Conmutación básica de tareas

Programación de Sistemas Operativos

David Alejandro González Márquez

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Introducción

Tareas

Introducción

Tareas

- Una *tarea* es una unidad de trabajo que el procesador puede **ejecutar**, **desalojar** e **intercambiar** por otra.

Tareas

- Una *tarea* es una unidad de trabajo que el procesador puede **ejecutar**, **desalojar** e **intercambiar** por otra.

- La tarea ejecuta una instancia de un programa y se puede ver como un **contexto de ejecución**.

Introducción

Tareas

- Una *tarea* es una unidad de trabajo que el procesador puede **ejecutar**, **desalojar** e **intercambiar** por otra.

- La tarea ejecuta una instancia de un programa y se puede ver como un **contexto de ejecución**.

- La arquitectura provee un mecanismo nativo para:
  - salvar el contexto de una tarea,
  - comenzarla a ejecutar o
  - conmutarla con otra.

Introducción

Una tarea está compuesta por:

Introducción

Una tarea está compuesta por:

1. Espacio de ejecución:
   - Segmento de código.
   - Segmento de datos/pila (uno o varios).
   - Mapa de páginas (CR3, con paginación esta activa).

Introducción

Una tarea está compuesta por:

1. Espacio de ejecución:
   - Segmento de código.
   - Segmento de datos/pila (uno o varios).
   - Mapa de páginas (CR3, con paginación esta activa).

2. Segmento de estado (TSS):
   - Almacena el estado de la tarea (su contexto) para poder reanudarla.
     - Registros de propósito general y selectores de segmento.
     - Flags, *CR3*, *LDTR* y Pilas de niveles de privilegio superior.

Introducción

Una tarea está compuesta por:

1. Espacio de ejecución: →**Memoria**
   - Segmento de código.
   - Segmento de datos/pila (uno o varios).
   - Mapa de páginas (CR3, con paginación esta activa).

2. Segmento de estado (TSS):
   - Almacena el estado de la tarea (su contexto) para poder reanudarla.
     - Registros de propósito general y selectores de segmento.
     - Flags, *CR3*, *LDTR* y Pilas de niveles de privilegio superior.

Introducción

Una tarea está compuesta por:

1. Espacio de ejecución: → **Memoria**
   - Segmento de código.
   - Segmento de datos/pila (uno o varios).
   - Mapa de páginas (CR3, con paginación esta activa).

2. Segmento de estado (TSS): → **Contexto**
   - Almacena el estado de la tarea (su contexto) para poder reanudarla.
     - Registros de propósito general y selectores de segmento.
     - Flags, *CR3*, *LDTR* y Pilas de niveles de privilegio superior.

Introducción: Estructura

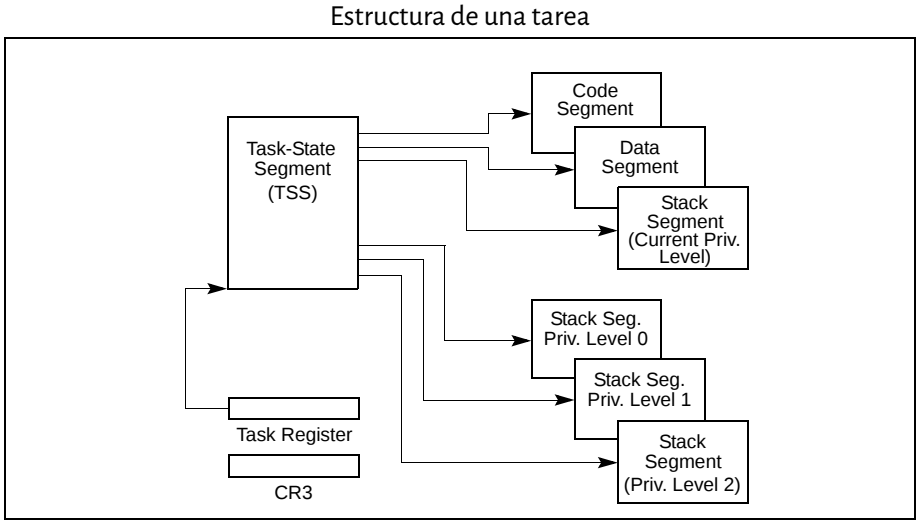


Figure 7-1. Structure of a Task

TSS: Task-State Segment

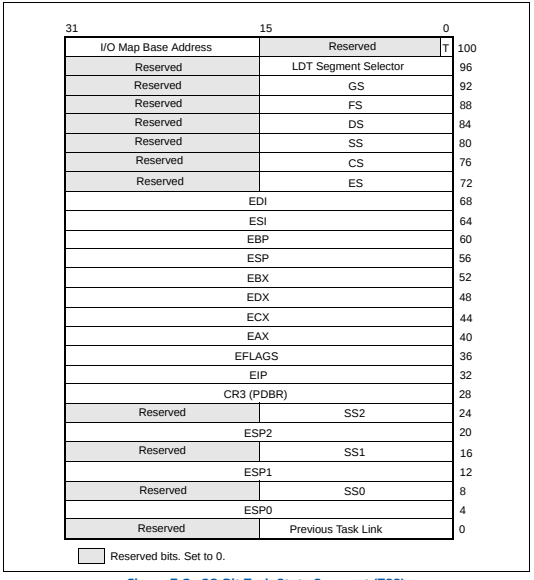


Figure 7-2. 32-Bit Task-State Segment (TSS)

TSS: Task-State Segment

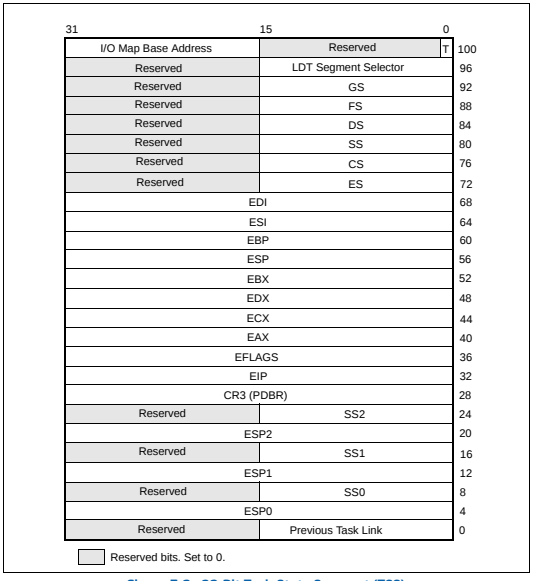


Figure 7-2. 32-Bit Task-State Segment (TSS)

- Una tarea está identificada por un selector de segmento de TSS.

TSS: Task-State Segment

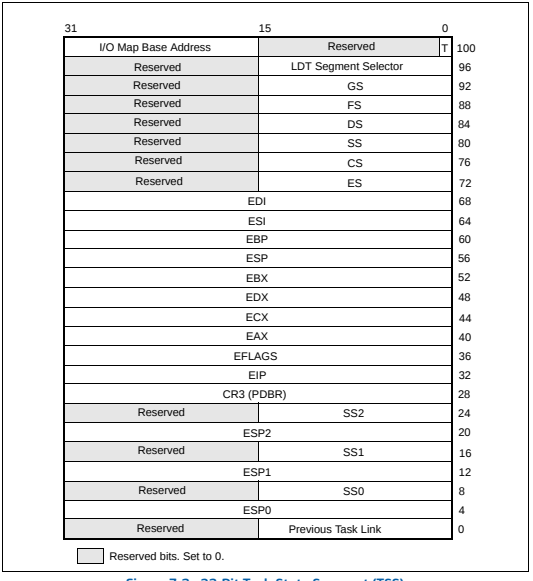


Figure 7-2. 32-Bit Task-State Segment (TSS)

- Una tarea está identificada por un selector de segmento de TSS.
- La TSS debe estar descripta en la GDT del mismo modo que se describen los segmentos de código y datos.

TSS: Task-State Segment

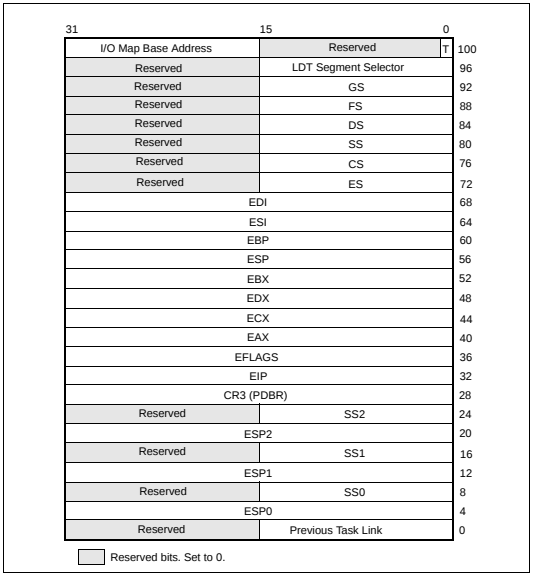


Figure 7-2. 32-Bit Task-State Segment (TSS)

- Una tarea está identificada por un selector de segmento de TSS.
- La TSS debe estar descripta en la GDT del mismo modo que se describen los segmentos de código y datos.
- El registro Task Register (TR) contiene selector de segmento de TSS de la tarea que se está ejecutando actualmente.

TSS: Descriptor de TSS

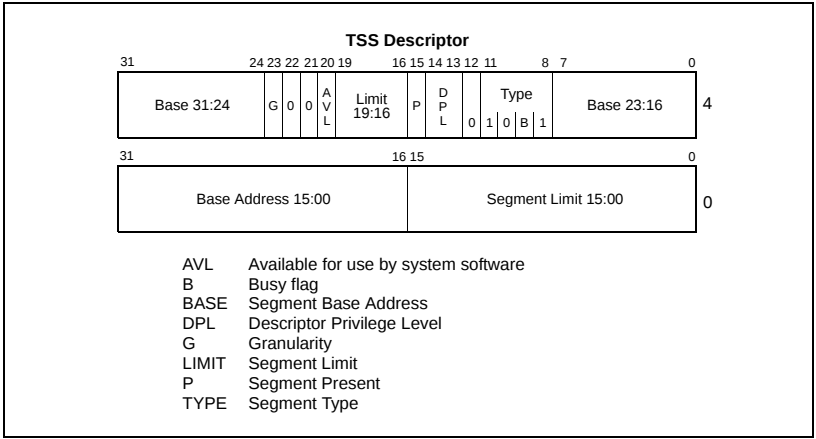


Figure 7-3. TSS Descriptor

TSS: Relación entre Task Register y GDT y TSS de la tarea acutal

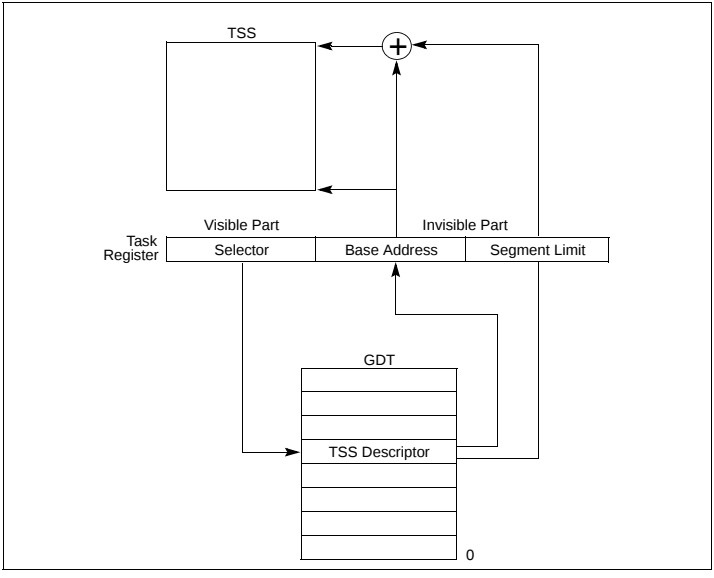
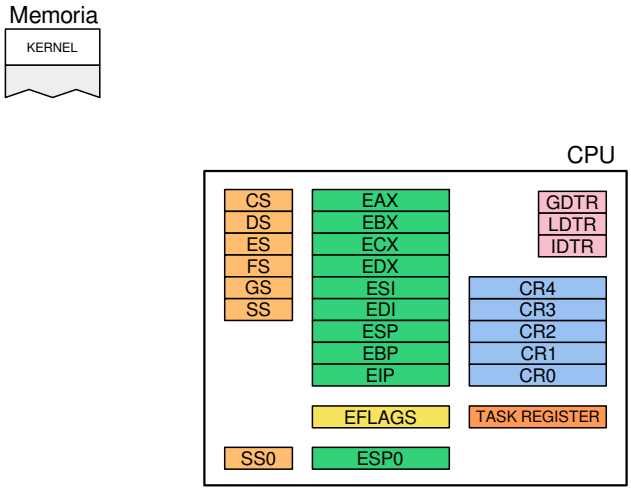
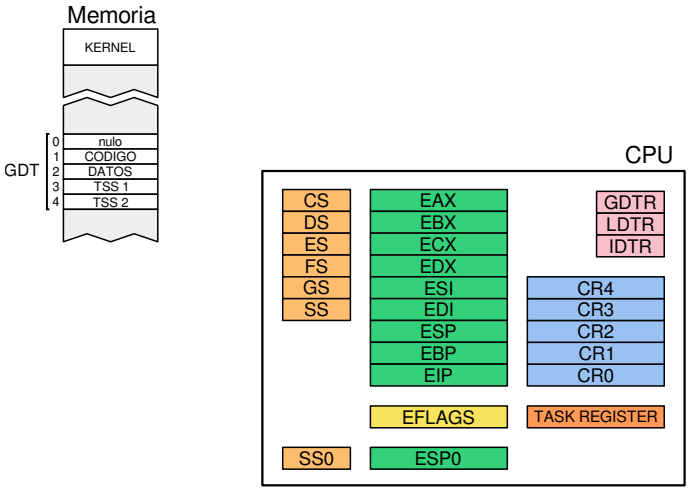


Figure 7-5. Task Register

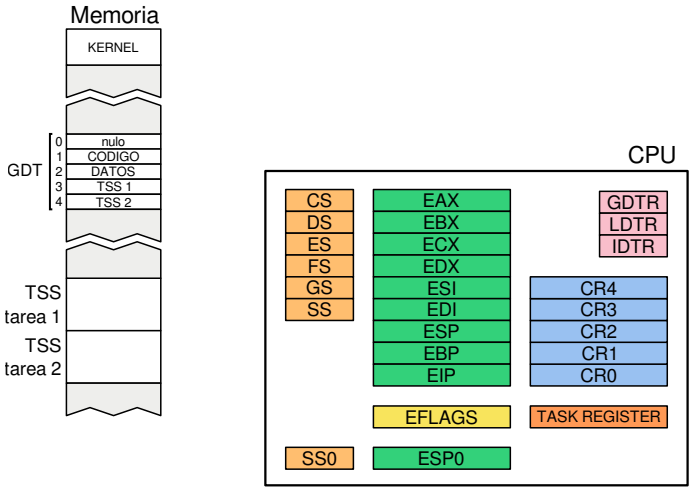
Intercambio de tareas



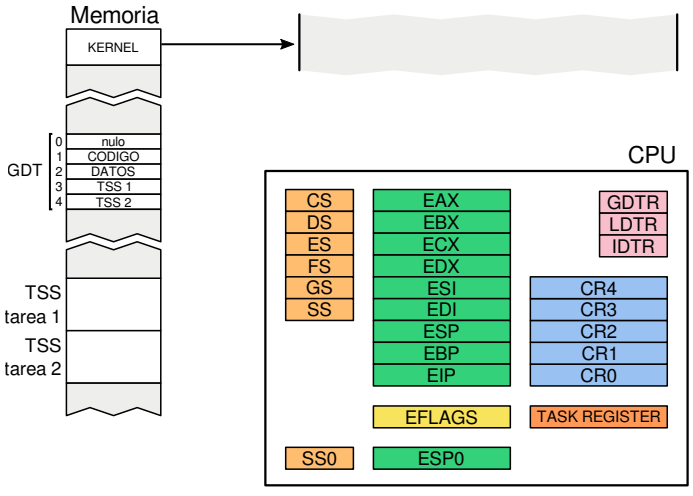
Intercambio de tareas



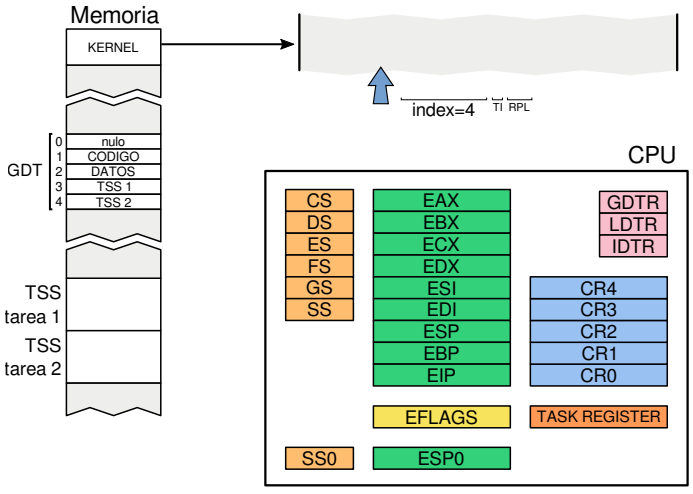
Intercambio de tareas



Intercambio de tareas

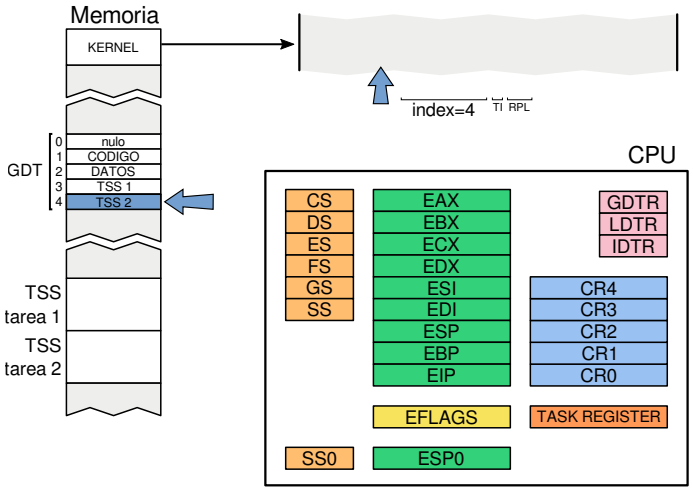


Intercambio de tareas



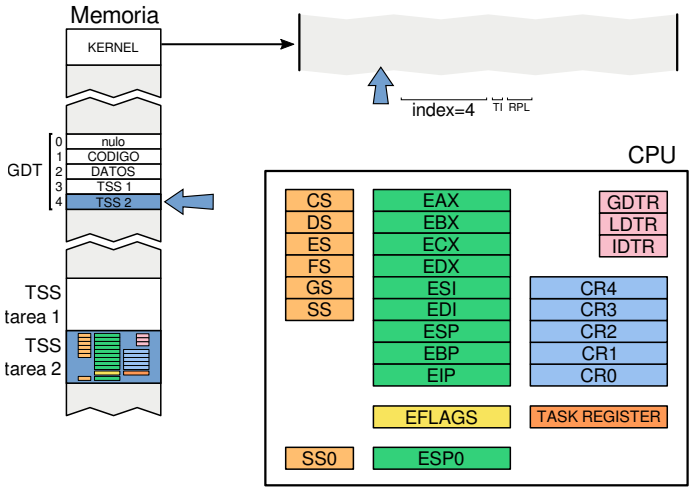
1. Determinar y validar TSS a ejecutar

Intercambio de tareas



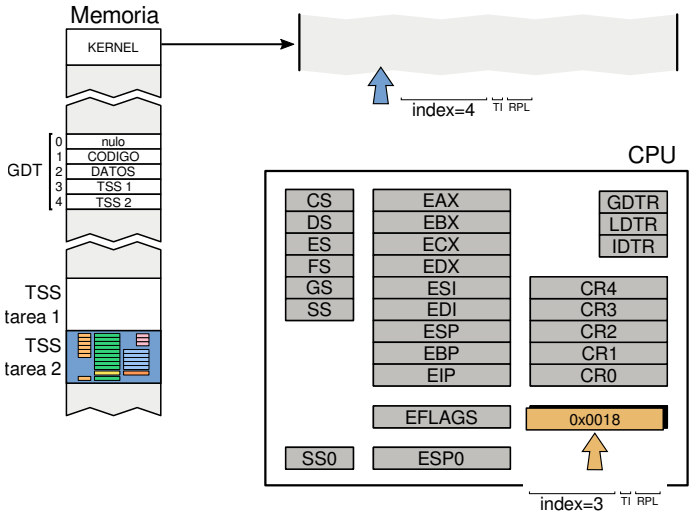
1. Determinar y validar TSS a ejecutar

Intercambio de tareas



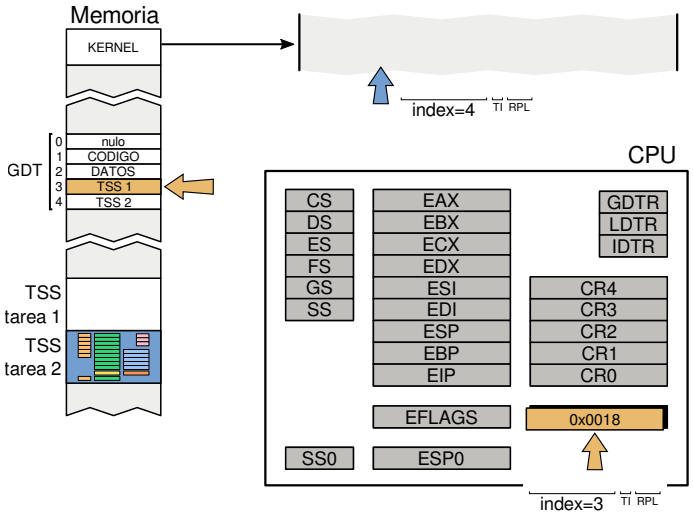
1. Determinar y validar TSS a ejecutar

Intercambio de tareas



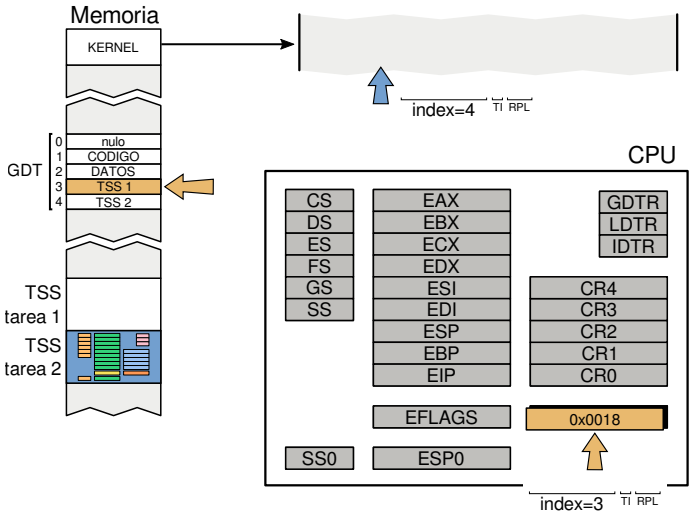
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución

Intercambio de tareas



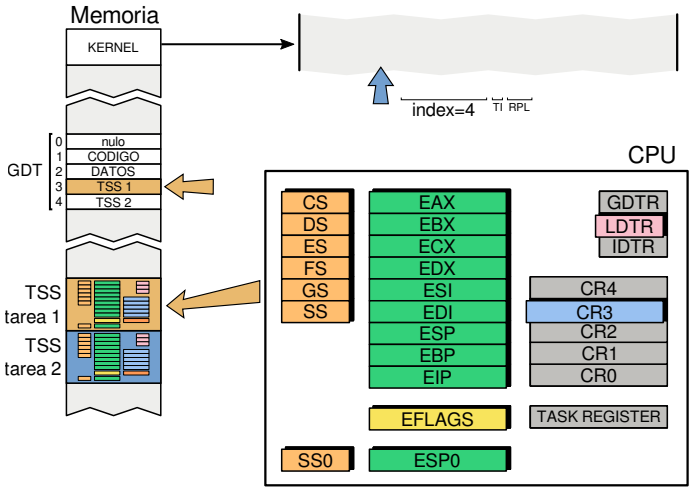
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución

Intercambio de tareas



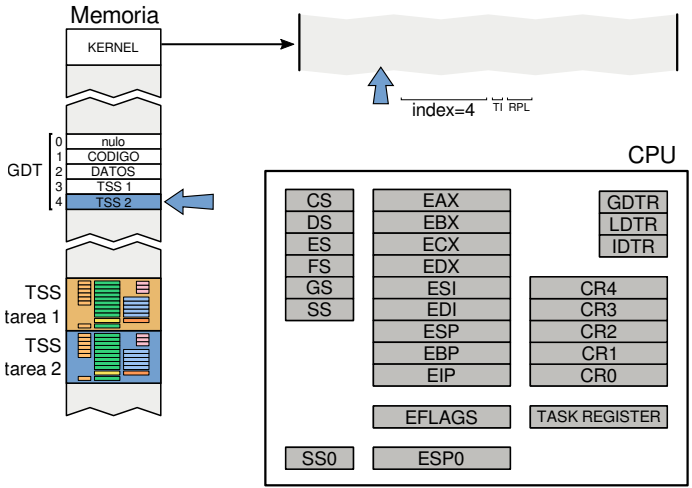
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria

Intercambio de tareas



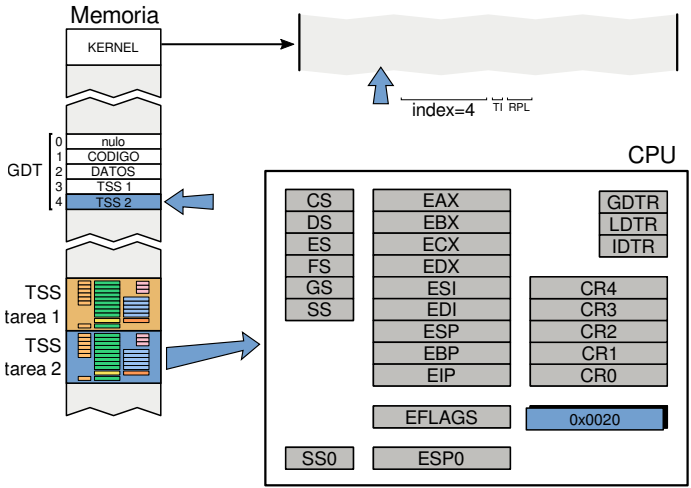
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria

Intercambio de tareas



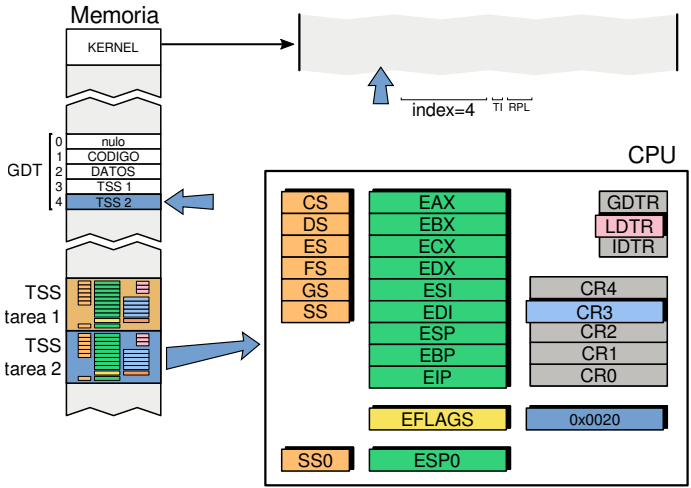
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria
4. Cargar el nuevo contexto de memoria

Intercambio de tareas



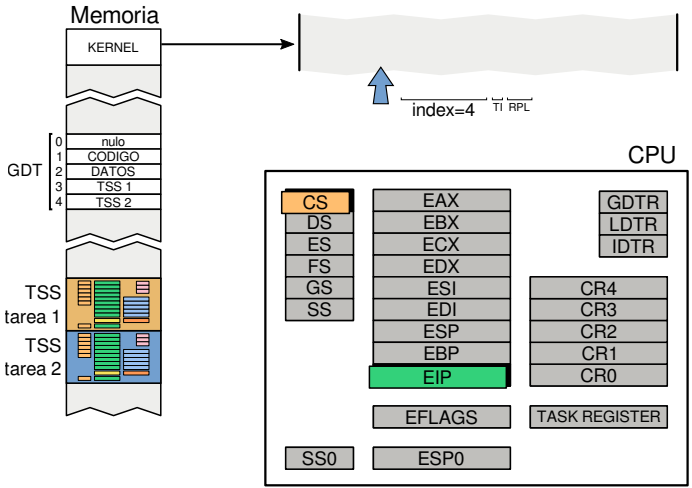
1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria
4. Cargar el nuevo contexto de memoria

Intercambio de tareas



1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria
4. Cargar el nuevo contexto de memoria

Intercambio de tareas



1. Determinar y validar TSS a ejecutar
2. Identificar la tarea en ejecución
3. Guardar el contexto actual a memoria
4. Cargar el nuevo contexto de memoria
5. Continuar la ejecución desde `CS:EIP`

Conmutación de Tareas: Tarea Inicial

Conmutación de Tareas: Tarea Inicial

1. Siempre que se salta a una tarea, hay un cambio de contexto, **¡Siempre!**

Conmutación de Tareas: Tarea Inicial

1. Siempre que se salta a una tarea, hay un cambio de contexto, **¡Siempre!**

2. El procesador guarda el contexto actual de la tarea (identificada en TR) y carga el contexto de la tarea a la cual se está saltando.

Conmutación de Tareas: Tarea Inicial

1. Siempre que se salta a una tarea, hay un cambio de contexto, **¡Siempre!**

2. El procesador guarda el contexto actual de la tarea (identificada en TR) y carga el contexto de la tarea a la cual se está saltando.

3. Entonces,
   ¿qué pasa la primera vez?
   ¿Qué pasa cuando se salta a la primera tarea?
   ¿Qué valor contiene TR?
   ¿Dónde se guarda el contexto?

Conmutación de Tareas: Tarea Inicial

1. Siempre que se salta a una tarea, hay un cambio de contexto, **¡Siempre!**

2. El procesador guarda el contexto actual de la tarea (identificada en TR) y carga el contexto de la tarea a la cual se está saltando.

3. Entonces,
   ¿qué pasa la primera vez?
   ¿Qué pasa cuando se salta a la primera tarea?
   ¿Qué valor contiene TR?
   ¿Dónde se guarda el contexto?

4. Hay que crear una **tarea inicial** para proveer una TSS en donde el procesador pueda guardar el contexto actual. Esta tarea inicial tiene este único propósito.

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, $\cdots$,**SS**

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, $\cdots$,**SS**
   - completar **CR3**

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, · · · ,**SS**
   - completar **CR3**
   - completar **EFLAGS**

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, · · · ,**SS**
   - completar **CR3**
   - completar **EFLAGS**

2. Al saltar por primera vez a una tarea:
   - tener un descriptor en la GDT de la tarea inicial

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, $\cdots$,**SS**
   - completar **CR3**
   - completar **EFLAGS**

2. Al saltar por primera vez a una tarea:
   - tener un descriptor en la GDT de la tarea inicial
   - tener un descriptor en la GDT de la tarea a saltar

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, $\cdots$,**SS**
   - completar **CR3**
   - completar **EFLAGS**

2. Al saltar por primera vez a una tarea:
   - tener un descriptor en la GDT de la tarea inicial
   - tener un descriptor en la GDT de la tarea a saltar
   - tener en **TR** algún valor válido (tarea inicial)

Conmutación de Tareas: Checklist

1. Al iniciar las tareas:
   - completar **EIP**
   - completar **ESP** y **EBP**
   - completar selectores de segmento **CS**, **DS**, · · · ,**SS** (recordar RPL)
   - completar **CR3**
   - completar **EFLAGS** (*)

2. Al saltar por primera vez a una tarea:
   - tener un descriptor en la GDT de la tarea inicial
   - tener un descriptor en la GDT de la tarea a saltar
   - tener en **TR** algún valor válido (tarea inicial)

Conmutación de Tareas: EFLAGS

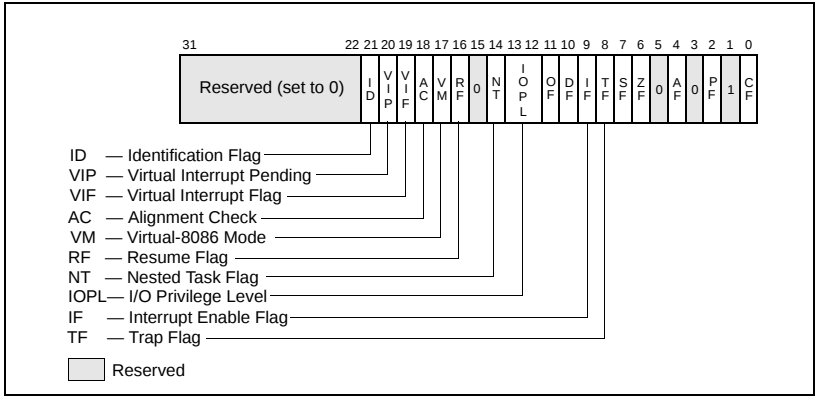


Figure 2-4. System Flags in the EFLAGS Register

EFLAGS por defecto

`0x00000002`

EFLAGS con Interrupciones habilitadas

`0x00000202`

Conmutación de Tareas: Instrucciones

- Cargar la tarea inicial en el TR

```
mov ax, <sel.init>
ltr ax
```

Conmutación de Tareas: Instrucciones

- Cargar la tarea inicial en el TR

```
mov ax, <sel.init>
ltr ax
```

- Saltar a la nueva tarea (intercambio)

```
jmp <sel.task>:0
```

Conmutación de Tareas: Instrucciones

- Cargar la tarea inicial en el TR

```
mov ax, <sel.init>
ltr ax
```

- Saltar a la nueva tarea (intercambio)

```
jmp <sel.task>:0
```

`<sel.init>`	**Tarea inicial**	Primera tarea que se carga, destino del contexto de ejecución actual.
`<sel.task>`	**Tarea a saltar**	Contexto de ejecución válido que se cargará en el procesador.

Trabajo Práctico

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.
   `DS` · · · `SS` = Selector de segmento de datos de nivel cero.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.
   `DS` $\cdots$ `SS` = Selector de segmento de datos de nivel cero.
   `EEFLAGS` = `0x202`. Interrupciones activas.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:
   - `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   - `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   - `ESP` = `0x27000`. Misma pila del Kernel.
   - `CS` = Selector de segmento de código de nivel cero.
   - `DS` · · · `SS` = Selector de segmento de datos de nivel cero.
   - `EEFLAGS` = `0x202`. Interrupciones activas.
3. Escribir el código necesario para ejecutar la tarea `Idle`, es decir, saltar intercambiando las TSS, entre la *tarea inicial* y la tarea `Idle`.

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.
   `DS` · · · `SS` = Selector de segmento de datos de nivel cero.
   `EEFLAGS` = `0x202`. Interrupciones activas.
3. Escribir el código necesario para ejecutar la tarea `Idle`, es decir, saltar intercambiando las TSS, entre la *tarea inicial* y la tarea `Idle`.

```
mov ax, <selector de segmento de la tarea inicial>
```

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.
   `DS` $\cdots$ `SS` = Selector de segmento de datos de nivel cero.
   `EEFLAGS` = `0x202`. Interrupciones activas.
3. Escribir el código necesario para ejecutar la tarea `Idle`, es decir, saltar intercambiando las TSS, entre la *tarea inicial* y la tarea `Idle`.

```
mov ax, <selector de segmento de la tarea inicial>
ltr ax
```

Trabajo Práctico

En el TP se debe crear una tarea inicial y luego saltar a la tarea *Idle*. Este ejercicio tiene el propósito de realizar un intercambio de tareas.

Pasos para saltar a la tarea *Idle*

1. Completar las entradas en la GDT para la *tarea inicial* y la tarea Idle.
2. Completar la TSS de la tarea Idle:

   `EIP` = `0x1A000`. Comienzo del código de la tarea Idle.
   `CR3` = `0x27000`. El mismo mapa de paginación del `kernel`.
   `ESP` = `0x27000`. Misma pila del Kernel.
   `CS` = Selector de segmento de código de nivel cero.
   `DS` · · · `SS` = Selector de segmento de datos de nivel cero.
   `EEFLAGS` = `0x202`. Interrupciones activas.
3. Escribir el código necesario para ejecutar la tarea `Idle`, es decir, saltar intercambiando las TSS, entre la *tarea inicial* y la tarea `Idle`.

```
mov ax, <selector de segmento de la tarea inicial>
ltr ax
jmp <selector de segmengo de la tarea Idle>:0
```

Bibliografía: Fuentes y material adicional

- Convenciones de llamados a función en x86:
  `https://en.wikipedia.org/wiki/X86_calling_conventions`
- Notas sobre System V ABI:
  `https://wiki.osdev.org/System_V_ABI`
- Documentación de NASM:
  `https://nasm.us/doc/`
- Artículo sobre el flag `-pie`:
  `https://eklitzke.org/position-independent-executables`
- Documentación de System V ABI:
  `https://uclibc.org/docs/psABI-x86_64.pdf`
- Manuales de Intel:
  `https://software.intel.com/en-us/articles/intel-sdm`

¡Gracias!

Recuerden leer los comentarios al final de este video por aclaraciones o fe de erratas.