

## Práctica 8 - Tareas

- Una tarea es una unidad de trabajo que el procesador puede despachar, ejecutar y suspender. Puede ser usada para ejecutar un programa.

→ Tareas

↳ Dos o más tareas distintas pueden tener un mismo código de programa; sin embargo, sus contextos de ejecución y datos asociados pueden ser distintos

↳ Podemos pensarlo como distintas instancias de un programa

En memoria una tarea va a tener:

① Espacio de ejecución: Es decir, páginas maestras donde va a tener el código, datos y píes. Podriremos precisar definirle un pd con sus pt o reutilizar algún directorio entre varias tareas.

② Segmento de Estado (TSS): Una región de memoria que almacena el estado de una tarea, se le espera de iniciarse o al momento de ser desalojada. Contiene:

- Registros de propósito general
- Registros de seg de la tarea
- Flags, seg aux nivel 0
- CR3, EIP

## Scheduler

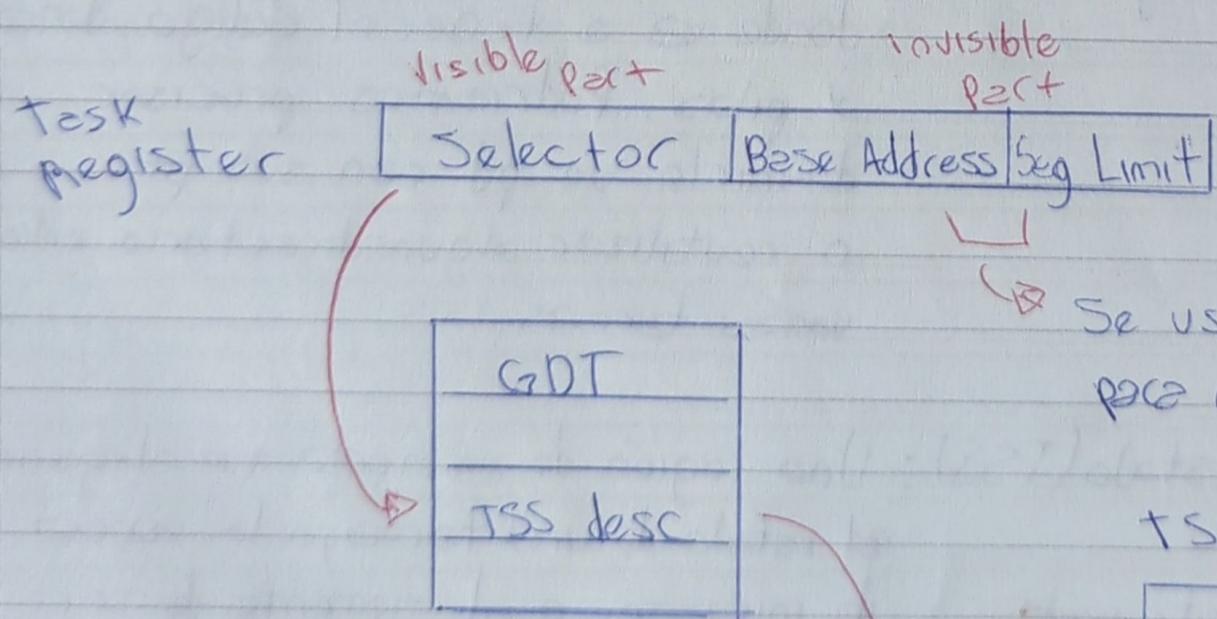
→ es un módulo de software que administra la ejecución de tareas/procesos. Utiliza una política o criterio para decir cuál es la proxima tarea a ejecutar.

→ Cada vez que pasa de una tarea a otra ocurre un cambio de contexto.

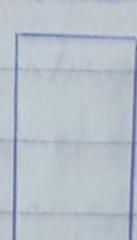
El procesador corre una única tarea por vez y cada tarea tiene su propio contexto de ejecución el cual habrá que cambiar de manera acorde

• Task register: almacena el selector de segmento de la tarea en ejecución

→ Se usa para encontrar la TSS actual



→ Se usan como cache para no tener que ir otra vez a la GOT



Para inicializar la TSS de una tarea, tenemos que completar con la info necesaria que posibilite la correcta ejecución de la tarea:

→ Campos mas relevantes

- ① EIP
- ② ESP, EBP y ESD
- ③ CR3
- ④ EFLAGS en 0x00000202 para tener las interrupciones habilitadas

### Descriptor de TSS

- Es una entrada en la GDT de tipo TSS descriptor
 

→ Cada TSS tiene que tener su correspondiente entrada en la GDT

### Atributos mas importantes

- El bit B(busy) indica si la tarea está siendo ejecutada. Lo iniciamos en 0.
- El bit DPL (descriptor privilege level) es el nivel de privilegio que se necesita para acceder al segmento. Usamos nivel 0 porque solo el kernel puede intercambiar tareas.

- Limit es el tamaño máximo de la TSS. 64h es el mínimo requerido.
- Base indica la dirección base de la TSS.

### Tarea inicial o idle

- El procesador siempre precisa estar ejecutando una tarea aun que esta no haga nada

Tarea inicial: se ejecuta al arrancar la computadora

Tarea idle: se ejecuta cuando no hay tareas para ejecutar.

LTR ax (ax = selector segmento tarea inicial)

JMP Selector Tarea Idle : 0

↓ offset, ignorado

→ produce  
cambio de  
contexto

## Context Switch

- El procesador se encuentra siempre ejecutando una tarea

→ La tarea actual está indicada por el registro  
TR (task register)

jmp 0x30:0

① En un tick del reloj el scheduler pide cambiar la tarea en ejecución haciendo `jmp` al segmento de la TSS de la nueva tarea.

② Antes de cambiar la tarea copia el estado actual de los registros en la TSS de la tarea de TR

→ TR → TSS Descriptor → TSS  
 selector (16 bits)

③ Antes de cambiar la tarea hay que restaurar el contexto de ejecución de la misma.

→ jmp 0x30:0  
 → TSS Descriptor → TSS

④ Cambia TR

## Rutina de atención de interrupciones de reloj

offset : dd 0

selector: dw 0

global \_ISR32 :

pushad ] preservo registros propósito general

call pic\_finish ] aviso que se ha atendido (es externo)

call sched\_nexttask ] trae en ax el selector de la tarea

stc cx ] cargo tr en un registro

cmp ax, cx ]

je .fin

↳ Si son la misma tarea  
sigue ejecutando

mov [selector], ax

jmp far [offset]

] salte al selector de TSS en la  
GDT y dispara el cambio de  
contexto

.fin:

popad

iret ] vuelve a la rutina que llamó  
restaurando EIP

## Nivel de privilegios en Tareas

stack segment  
↑

Caso 1 : Tarea ejecutando en nivel 0 indicado por su SS y se produce una interrupción de reloj. El nivel de ejecución no cambia dado que la interrupción de reloj es de nivel 0.

	← ESP antes de transferir el handler
EFLAGS	
CS	
EIP	
Error Code	← ESP después de transferir el handler

Caso 2: Tenemos una tarea de nivel 3 indicado por su SS y se produce una interrupción de reloj, el nivel de ejecución cambia.

Por lo tanto usa la pila de nivel 0 (ss01 indicada en la TSS para guardar la info de retorno)

Stack tarea interrumpida

← ESP antes de transferir el handler

ESP DESPUES  
de transferir →

SS	
ESP	
EFLAGS	
CS	
EIP	
ER CODE	

Si se produce un cambio de contexto, la TSS de una tarea de nivel 3 podría quedar con un SS almacenado de nivel 0.

↳ los valores nivel 3 quedan en la pila y se restauración en el IRET correspondiente.