

Familia xx86. Modo protegido

Protección de memoria.

A partir del 80286 aparece un nuevo modo de funcionamiento: **Modo protegido**.

Se pretende establecer mecanismos de control para evitar que desde una tarea (que funcione de forma errática o diseñada de forma mal intencionada) se pueda acceder a zonas de memoria pertenecientes al Sistema Operativo o a otras tareas.

Este objetivo se consigue apoyándose en 2 pilares.

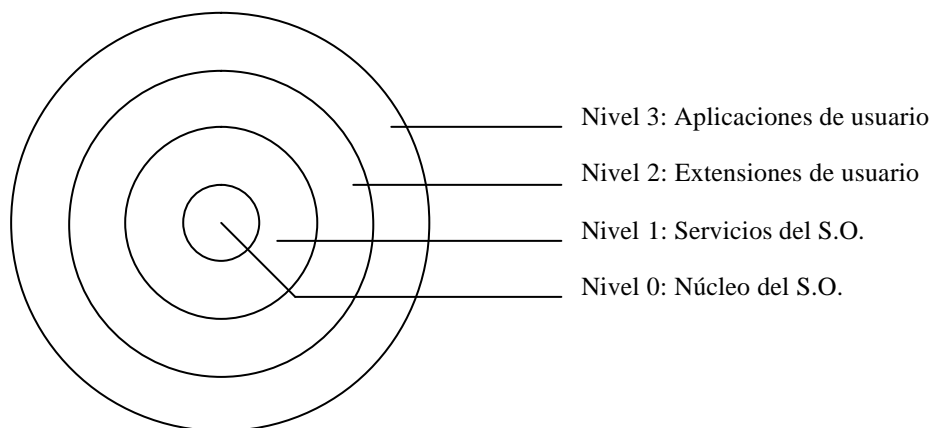
✂ **Memoria segmentada.**

✂ **Niveles de privilegio.**

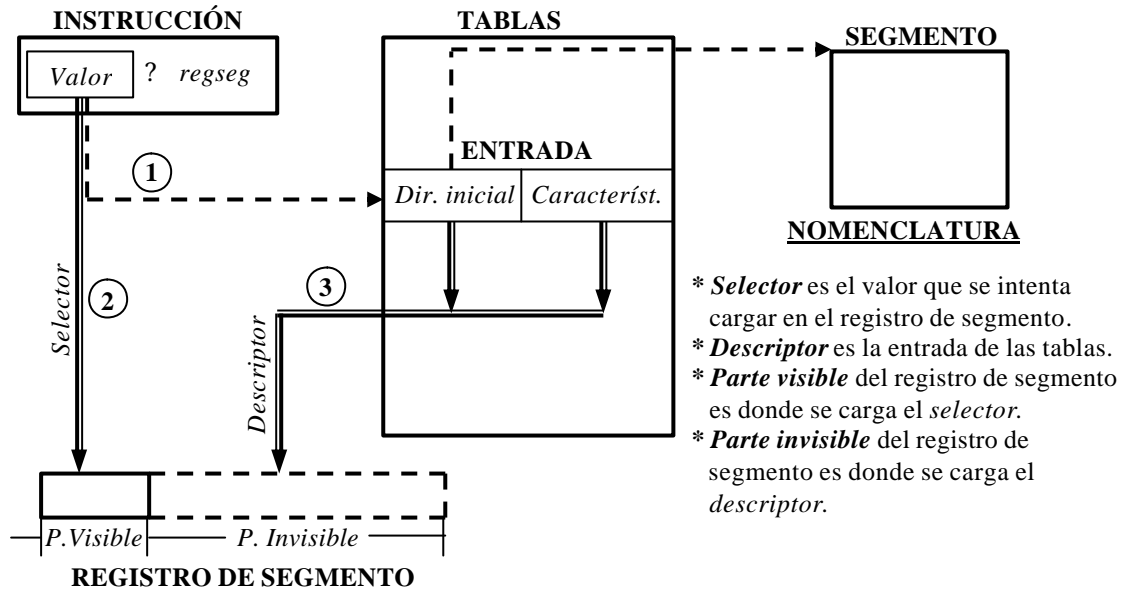
Memoria segmentada: Potenciar el sistema de acceso a memoria a través de registros de segmento. El registro de segmento no solo indica donde comienza un segmento, sino que además indica: tamaño, nivel de privilegio necesario, tipos de acceso permitidos, ...

En un instante dado solo son accesibles las zonas de memoria que estén apuntadas por un registro de segmento. Se establece un control completo a todas las acciones encaminadas a cargar uno de estos registros.

Niveles de privilegio: Se establecen 4 niveles de privilegio. numerados de 0 a 3. Se asigna a las tareas de usuario el nivel menos privilegiado (3), y al núcleo del S.O. el más privilegiado (0). En la figura se muestran dichos niveles como círculos concéntricos, y un ejemplo de uso utilizado por intel.



1 Carga directa de un registro de segmento.



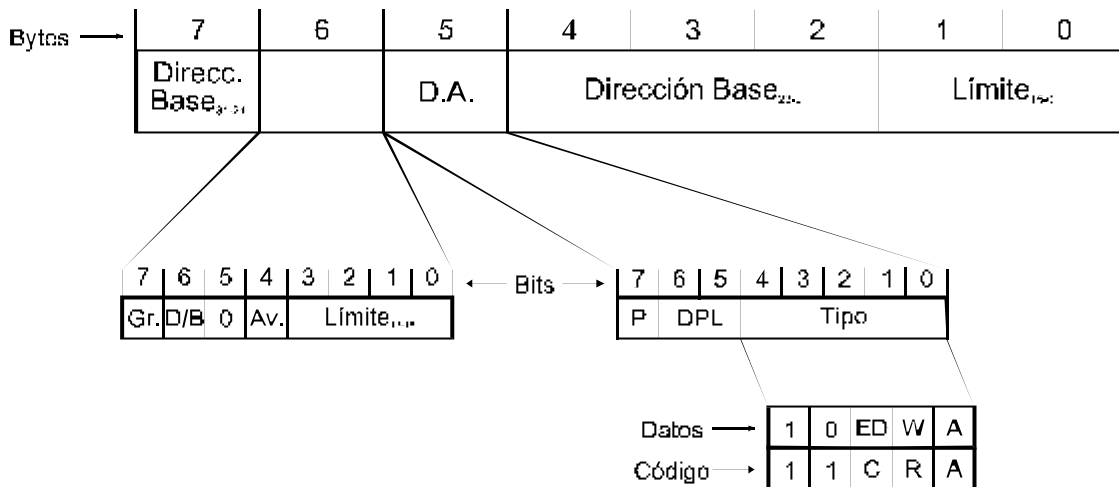
2 Accesos posibles a memoria .

TIPO DE DESCRIPTOR DE SEGMENTO					
		De código		De datos	
		Legible	Ilegible	Modificable	No modificable
TIPO DE ACCESO	Fetch	Sí	Sí	No	No
	Lectura	Sí	No	Sí	Sí
	Escritura	No	No	Sí	No

3 Resumen de offsets y tamaños de un segmento .

Características			OFFSETS	
ED	G	B	Inicial	Final
0	0	X	0	Lím.
0	1	X	0	Lím*1000H+0FFFH
1	0	0	Lím+1	0FFFFH
1	1	1	(Lím+1)*1000H	0FFFFFFFH

4 Estructura de un descriptor de segmento.



- ? **A** (Accessed): El micro lo pone a 1 cada vez que se accede al descriptor.
- ? **Av** (Available): Disponible para el S.O.
- ? **C** (Conforming): Si está a 1, el segmento no impone su privilegio como nuevo CPL al cargarse, y “se conforma” con el que hubiera anteriormente.
- ? **D/B** (Default/Big): En segmentos de código funciona como D (solo es activo si el descriptor se carga en CS), y si es 1, se direccionan con EIP. También define que se usarán por defecto offsets y operandos de 32 bits excepto en lo que afecta al direccionamiento de la pila. En segmentos de datos funciona como B, solo es significativo si el descriptor se carga en SS e indica, si es 1, que la pila se direccionará con ESP.
- ? **Dirección Base**: Valor de 32 bits que apunta al principio del segmento apuntado por el descriptor.
- ? **DPL** (Descriptor Privilege Level): Nivel de privilegio del segmento.
- ? **Gr.** (Granularity) y **Límite**: Especifican el offset límite del segmento. Si Gr. es 0, el límite de 20 bits especificado en el descriptor es directamente el offset límite, y si es 1, hay que añadirle a la derecha 12 bits a 1 (FFFH).
- ? **ED** (Expand Down): Si es 1 el segmento comienza en el offset límite + 1 y termina en el máximo posible (0FFFFH ó = 0FFFFFFFFH dependiendo de que B sea 0 ó 1 respectivamente).
- ? **P** (Present): Si es 0 indica que el segmento no está presente en memoria o que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **R** (Readable): Solo es activo para segmentos de código. Si es 1 se permiten accesos de lectura.
- ? **W** (Writable): Solo es activo para segmentos de datos. Si es 1 se permiten accesos de escritura.

NOTA: Obsérvese que la estructura del descriptor está un poco fragmentada (Dirección base y límite). Esto se debe a que el modo protegido surgió con el 80286, microprocesador con un bus de direcciones de 24 bits y de datos de 16 bits. En este entorno se definieron descriptores de 8 Bytes, pero los dos bytes de mayor peso se dejaron como reservados y los 6 restantes contenían la estructura que se muestra. Al aparecer el 80386 con bus de direcciones y datos de 32 bits, se procedió a utilizar los dos bytes reservados que como se puede observar contienen información relativa a este crecimiento a 32 bits.

5 Tablas de descriptores, registros relacionados y Selectores.

Los descriptores se almacenan en las tablas de descriptores. Una tabla de descriptores está limitada a un tamaño máximo de 64 KB. que permite almacenar 8K descriptores de 8 Bytes.

Existen tres tipos de tablas que contienen descriptores:

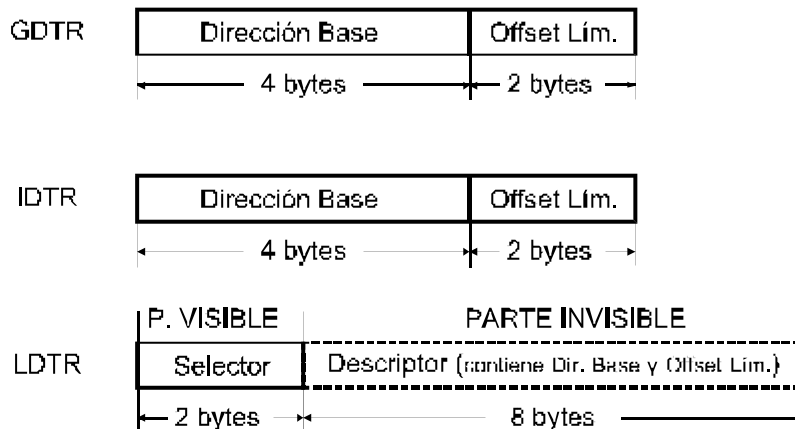
Tabla Global de descriptores. Normalmente existe una única instancia en memoria.

Tabla Local de descriptores. Normalmente existen varias instancias en memoria (una por cada tarea en curso), aunque en un momento dado solo hay una activa.

Tabla de descriptores de interrupción. Normalmente existe una única instancia en memoria

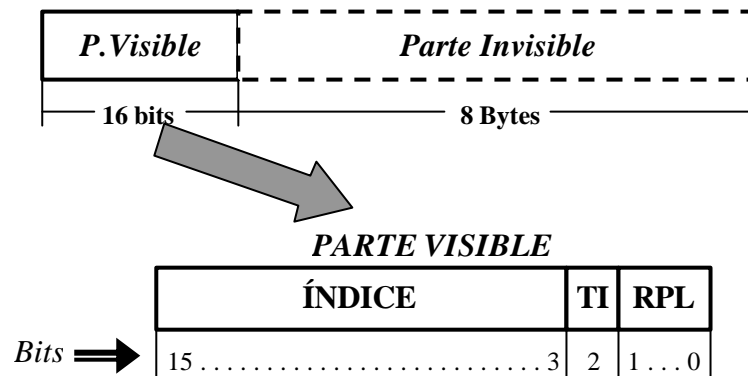
La CPU dispone de tres registros para apuntar a los tres tipos de tablas:

GDTR (Global Descriptor Table Register), **IDTR** (Interrupt Descriptor Table Register) y **LDTR** (Local Descriptor Table Register). Su estructura se muestra a continuación.



Selector: Es un entero de 16 bits que se almacena en la parte visible de los registros de segmento. Consta de 3 campos:

- ? **RPL** (Request Privilege Level): Nivel de privilegio requerido, indica cual es el nivel de privilegio que se quiere usar cuando se quiere cargar un descriptor.
- ? **TI** (Table Indicator): Indica en que tabla se quiere localizar un descriptor. Si vale 0 se hace referencia a la GDT, si vale 1 a la LDT. Los descriptors de la IDT no se apuntan mediante un selector, sino mediante un vector de interrupción.
- ? **Índice:** Indica el nº de descriptor dentro de la tabla. Como cada descriptor ocupa 8 bytes, el índice se multiplica por 8 para calcular el offset que le corresponde a un descriptor dentro de su tabla.

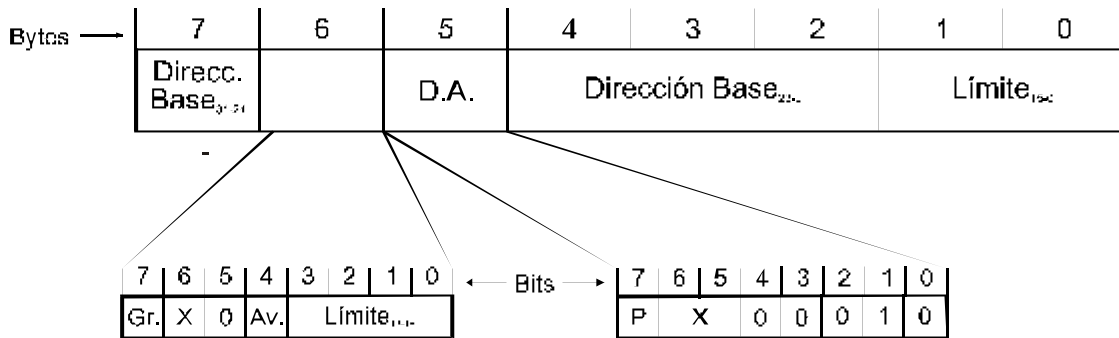


6 Tipos de descriptors .

Existen 8 tipos de descriptors. No todos los descriptors pueden almacenarse en todas las tablas. A continuación se muestran los diferentes tipos de descriptors, y las tablas en las que pueden aparecer.

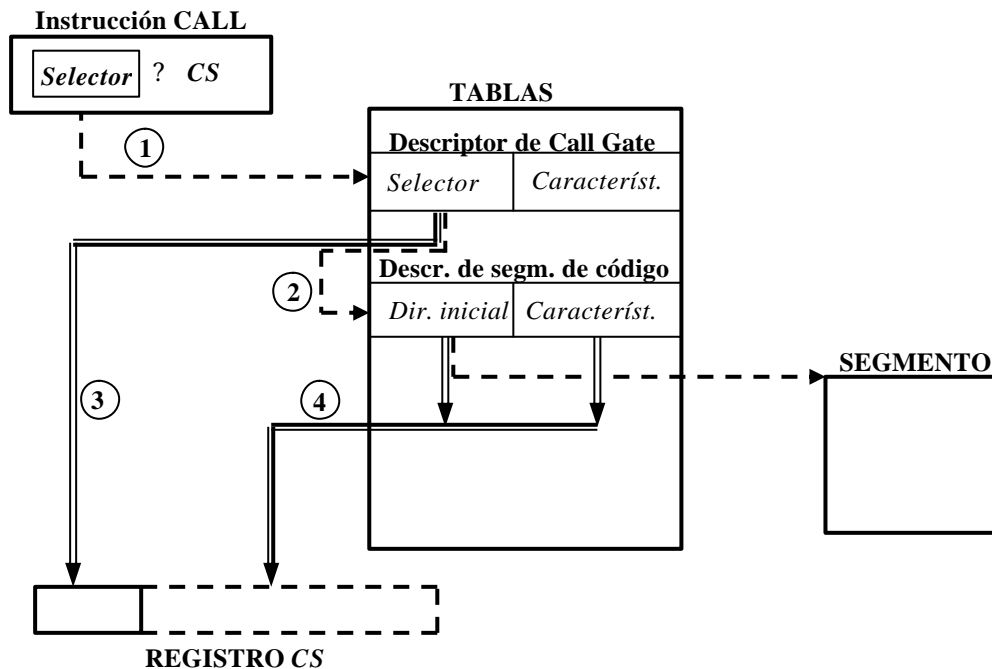
Tipo de descriptor	Descripción	GDT	LDT	IDT
Segmento de código	Apunta a un segmento de código	X	X	
Segmento de datos	Apunta a un segmento de datos	X	X	
TSS	Apunta a un segmento de estado de tarea	X		
LDT	Apunta a una tabla LDT	X		
CALL GATE	Descriptor que apunta a su vez a otro descriptor de segmento de código.	X	X	
INTERRUPTION GATE	Descriptor que apunta a su vez a otro descriptor de segmento de código.			X
TRAP GATE	Descriptor que apunta a su vez a otro descriptor de segmento de código.			X
TASK GATE	Descriptor que apunta a su vez a otro descriptor de tipo TSS.	X	X	X

7 Estructura de un descriptor de LDT.

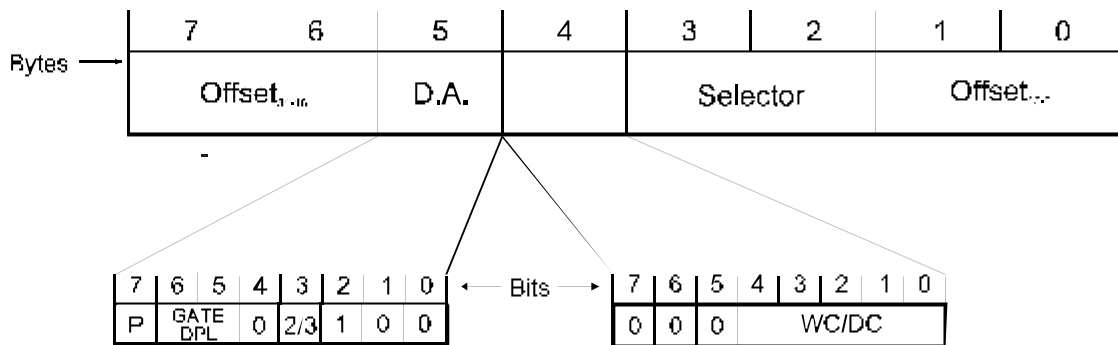


- ? **Av** (Available): Disponible para el S.O.
- ? **Dirección Base**: Valor de 32 bits que apunta al principio del segmento apuntado por el descriptor.
- ? **Gr.** (Granularity) y **Límite**: Especifican el offset límite del segmento. Si Gr. es 0, el límite de 20 bits especificado en el descriptor es directamente el offset límite, y si es 1, hay que añadirle a la derecha 12 bits a 1 (FFFH).
- ? **P** (Present): Si es 0 indica que el segmento no está presente en memoria o que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **X** Sin significado

8 Carga de CS mediante CALL GATE.

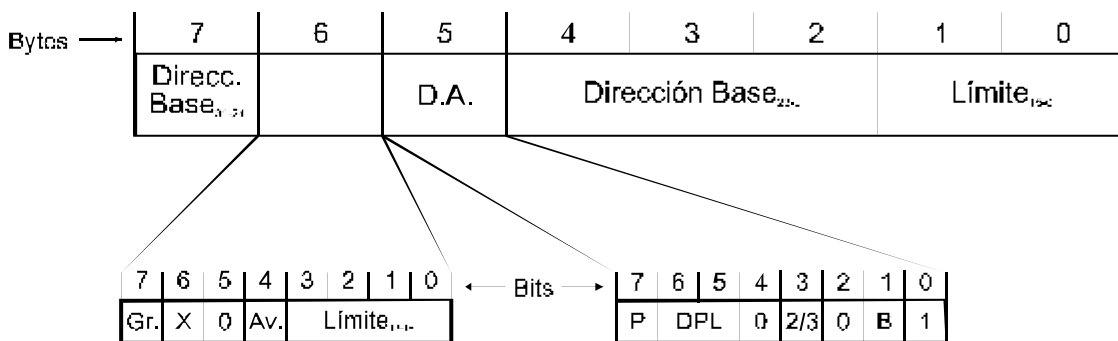


9 Descriptor de tipo CALL GATE.



- ? **2/3** (80286/80386): Si es 0, apunta a una *gate* de tipo 80286, que trata la pila a nivel de words. Si es 1, apunta a una de tipo 80386, que la manejan por *double words*.
- ? **Gate DPL** (*Descriptor Privilege Level*): Nivel de privilegio de la *gate*.
- ? **Offset**: Punto de entrada en la rutina.
- ? **P** (*Present*): Si es 0 indica que el segmento no está presente en memoria o que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **Selector**: Apunta al descriptor de segmento de código de la rutina. El campo RPL no se usa.
- ? **WC/DC**: (*Word count/Double word count*) Número de parámetros (*words* en caso de ser de 286, y *Double words* en caso de ser de 386) que se copian de una pila a otra en caso de que cambie el CPL.

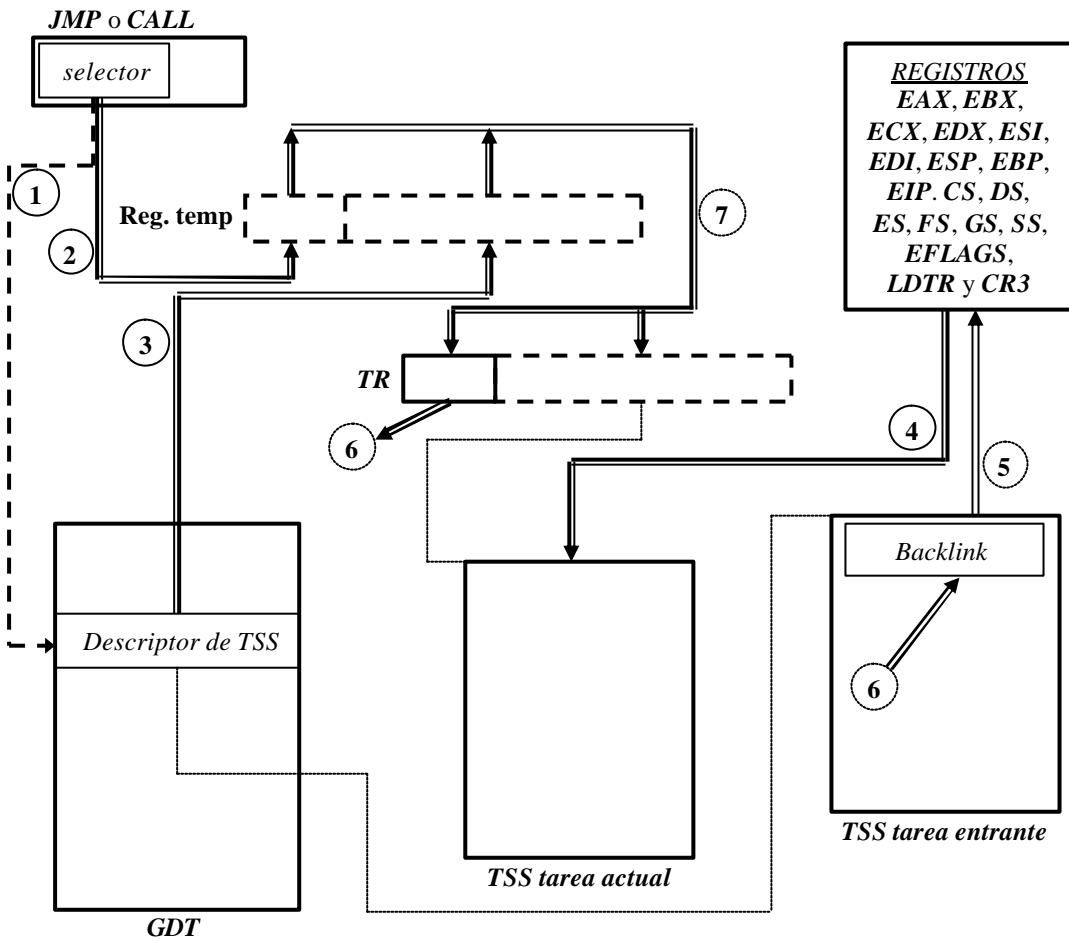
10 Descriptor de tipo TSS.



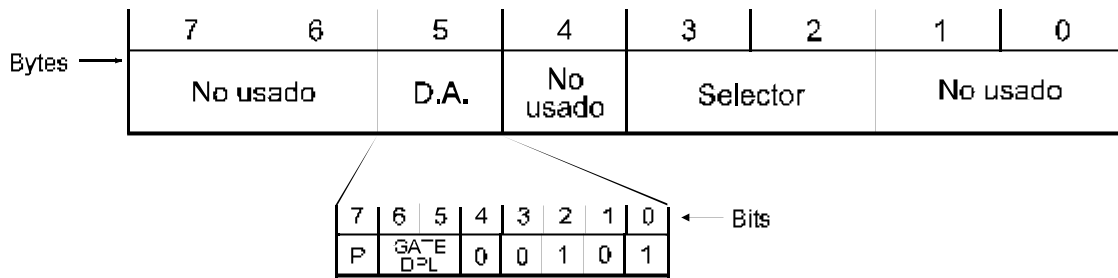
- ? **2/3** (80286/80386): Si es 0, apunta a una *gate* de tipo 80286, que trata la pila a nivel de words. Si es 1, apunta a una de tipo 80386, que la manejan por *double words*.
- ? **Av** (*Available*): Disponible para el S.O.
- ? **B** (*Busy*): Si está a '1' indica que la tarea está ocupada, y que no se la puede llamar ni saltar a ella. Están ocupadas la tarea actual y las que hubieran llamado a una tarea hija, porque están esperando su retorno.
- ? **Dirección Base**: Valor de 32 bits que apunta al principio del segmento apuntado por el descriptor.
- ? **DPL** (*Descriptor Privilege Level*): Nivel de privilegio del descriptor de TSS.

- ? **Gr.** (Granularity) y **Límite**: Especifican el offset límite del segmento. Si Gr. es 0, el límite de 20 bits especificado en el descriptor es directamente el offset límite, y si es 1, hay que añadirle a la derecha 12 bits a 1 (FFFH).
- ? **P** (Present): Si es 0 indica que el segmento no está presente en memoria o que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **X**: Sin significado.

11 Proceso de cambio de tarea directo.

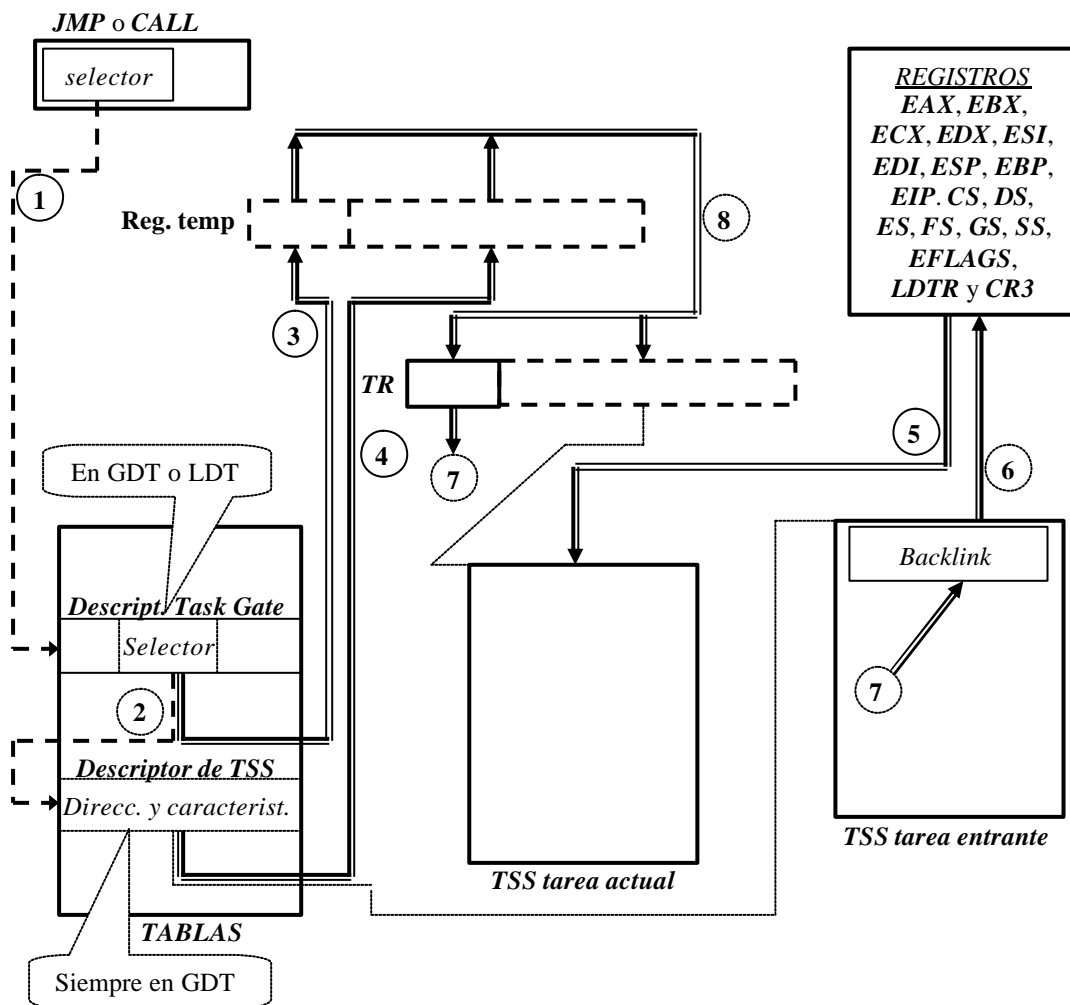


12 Descriptor de tipo TASK GATE.

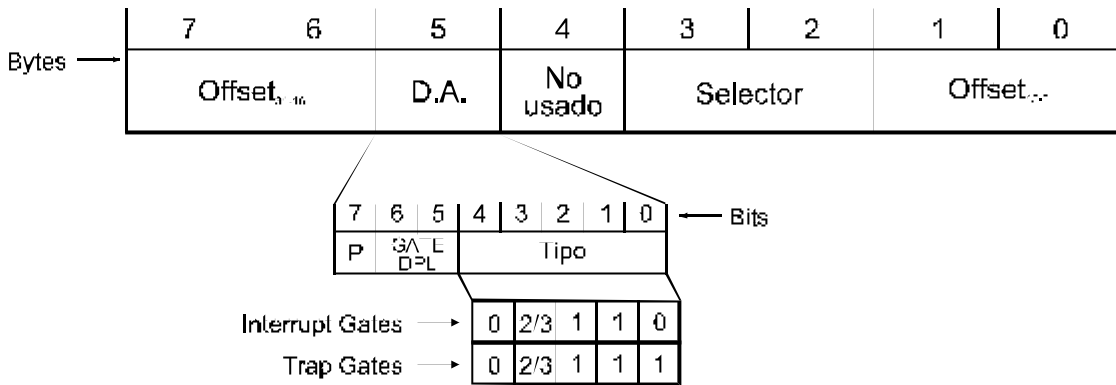


- ? **Gate DPL (Descriptor Privilege Level):** Nivel de privilegio de la *gate*.
- ? **P (Present):** Si es 0 indica que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **Selector:** Apunta al descriptor de TSS de código de la rutina. El campo RPL no se usa.

13 Proceso de cambio de tarea a través de TASK GATE.

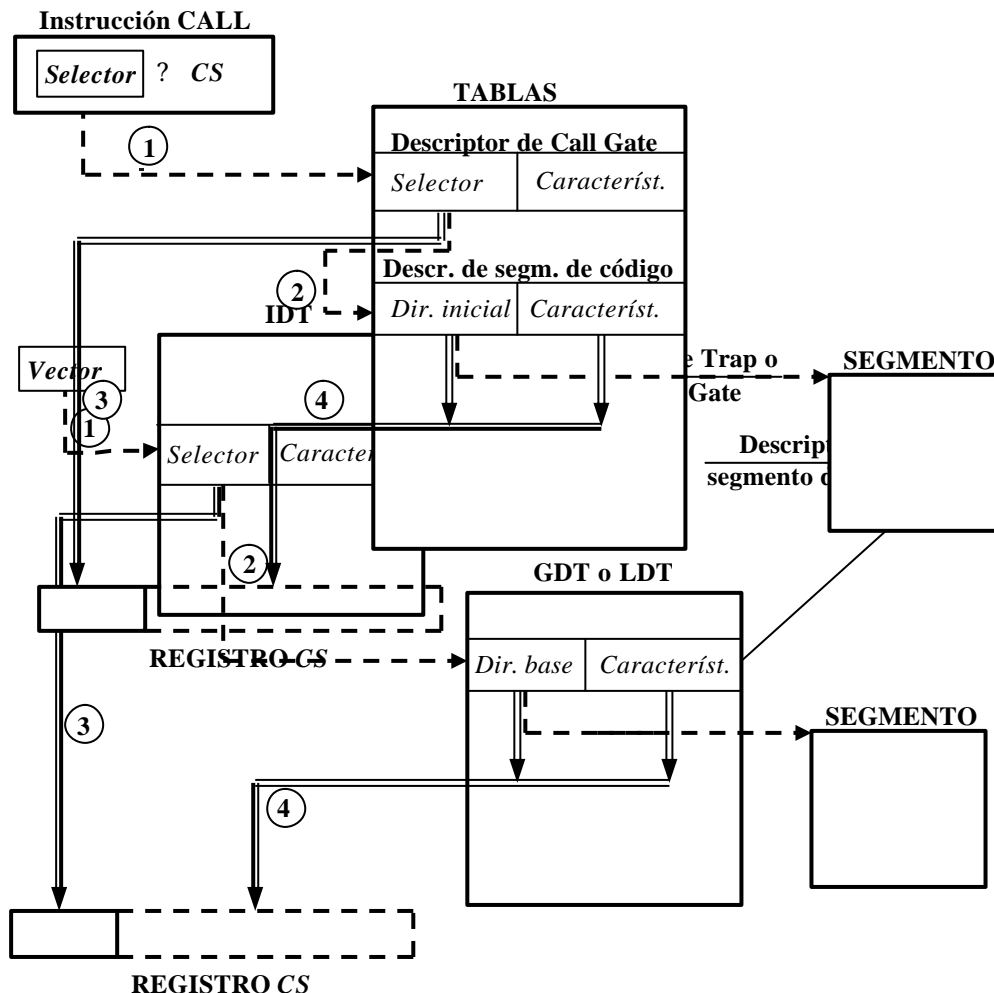


14 Descriptor de tipo Interrupt y Trap GATE.



- ? **2/3 (80286/80386):** Si es 0, apunta a una *gate* de tipo 80286, que trata la pila a nivel de words. Si es 1, apunta a una de tipo 80386, que la manejan por *double words*.
- ? **Gate DPL (Descriptor Privilege Level):** Nivel de privilegio de la *gate*.
- ? **Offset:** Punto de entrada en la rutina.
- ? **P (Present):** Si es 0 indica que la entrada de la tabla está vacía. El acceso a un descriptor con P=0 provoca una excepción con vector 0BH.
- ? **Selector:** Apunta al descriptor de segmento de código de la rutina. El campo RPL no se usa.

15 Carga de CS mediante Interrupt o Trap GATE.



16 Cambio de tarea mediante IRET con $F_{NT}=1$.

