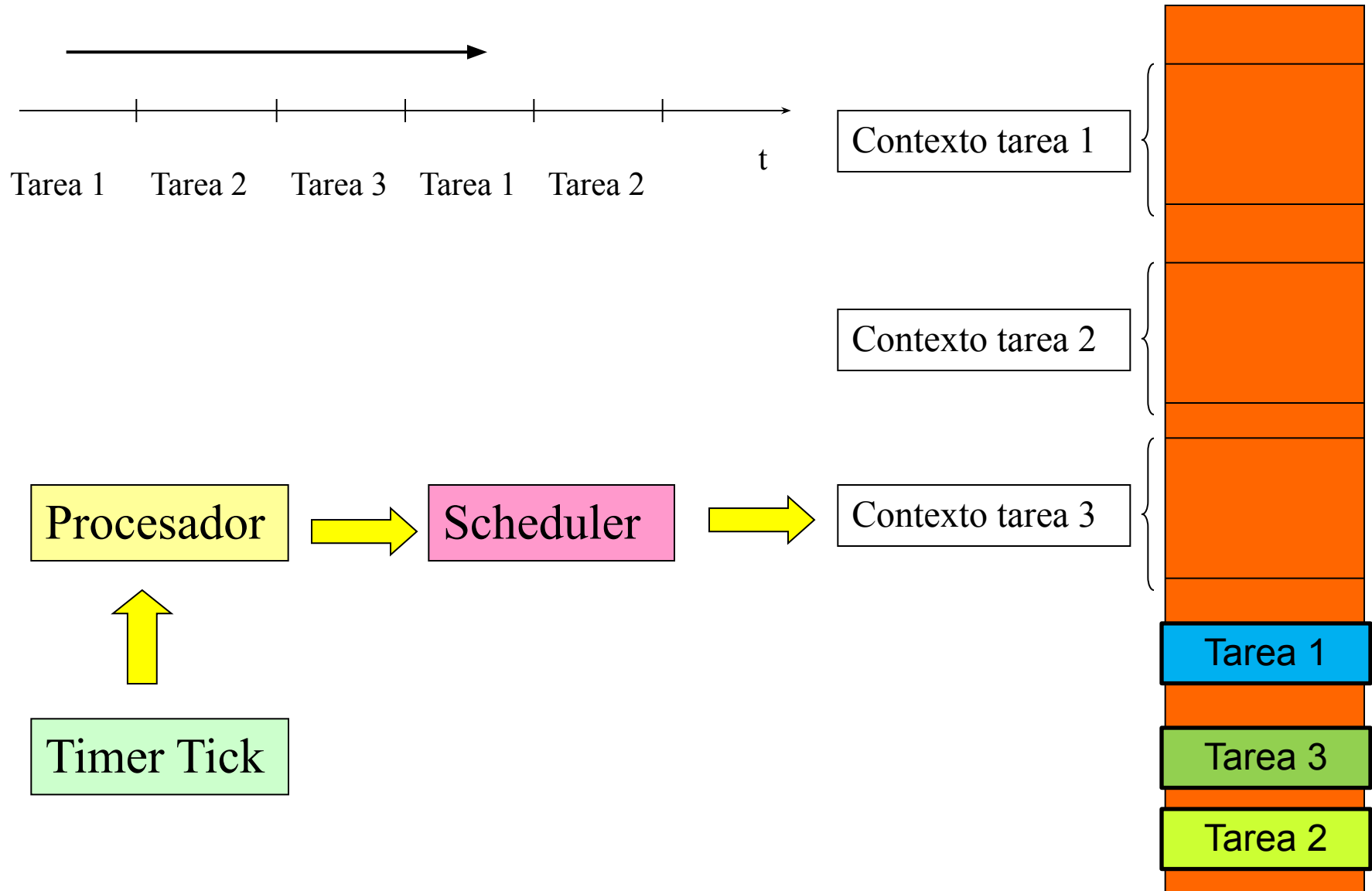


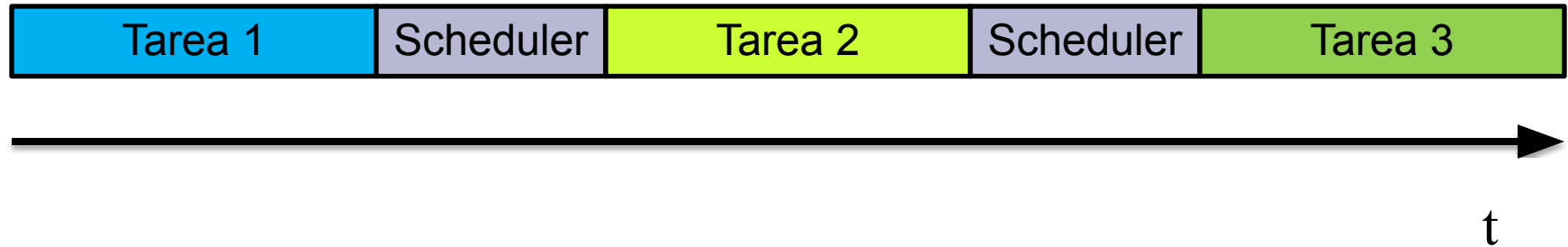


# Modo Protegido

# Modo protegido-Conmutación de tareas



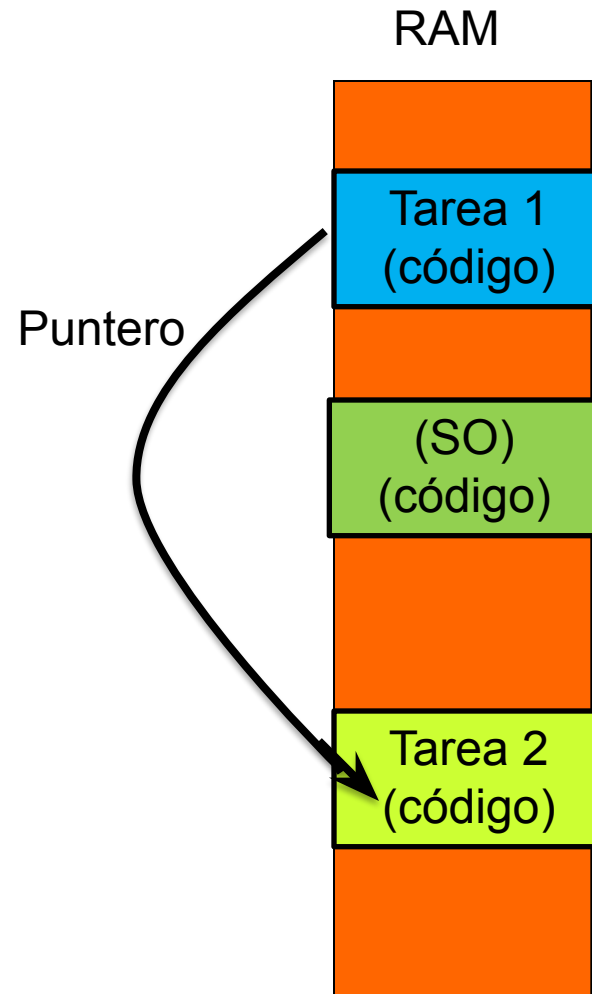
# Modo protegido-Conmutación de tareas



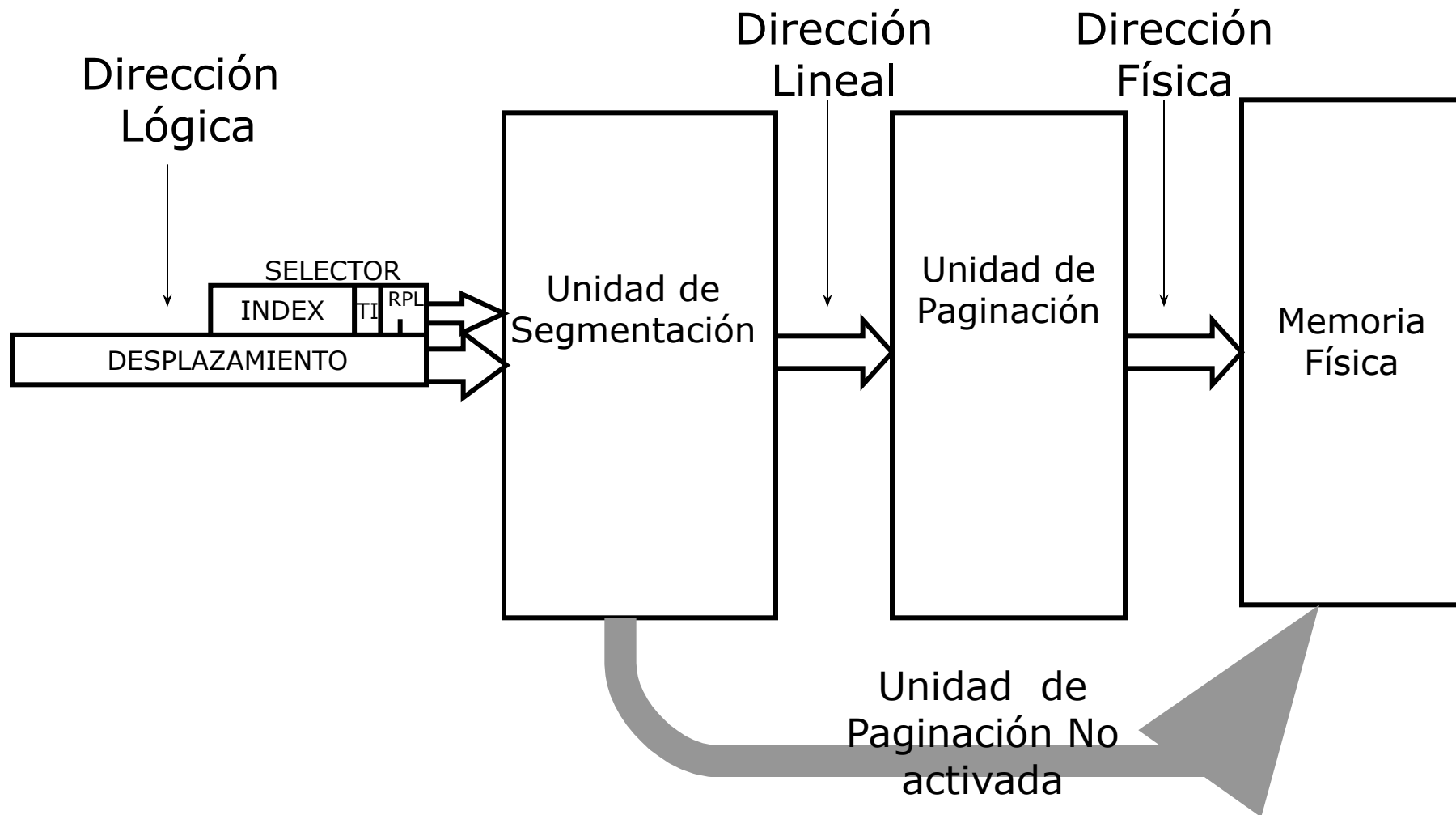
¿El SO es una tarea ?

# Modo protegido- Protección de tareas

¿Como puede evitar el SO  
este acceso?



# Modo protegido-MMU



# Modo protegido-MMU

- La unidad de segmentación **NO** se puede deshabilitar.
- La unidad de paginación **SI** se puede deshabilitar.

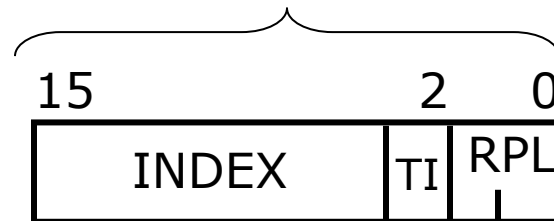
*Debido a esta limitación de los procesadores Intel, algunos sistemas operativos eligen utilizar la memoria en modo “flat” ( se verá luego) para no utilizar las reglas de segmentación pero si las de paginación.*

# Dirección lógica

Ejemplo: DS = 20h y Offset = 1000h

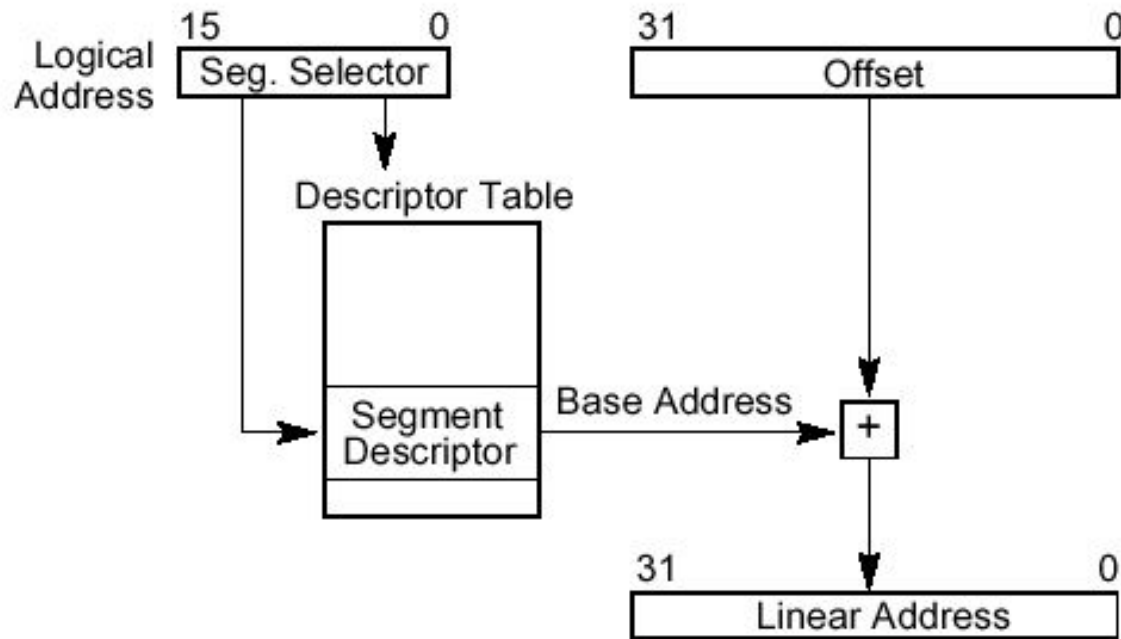
Mov DS:[1000h], EAX

SELECTOR



REGISTRO DE SEGMENTO  
(CS, DS, ES, SS, FS, o GS)

# Dirección lógica a lineal

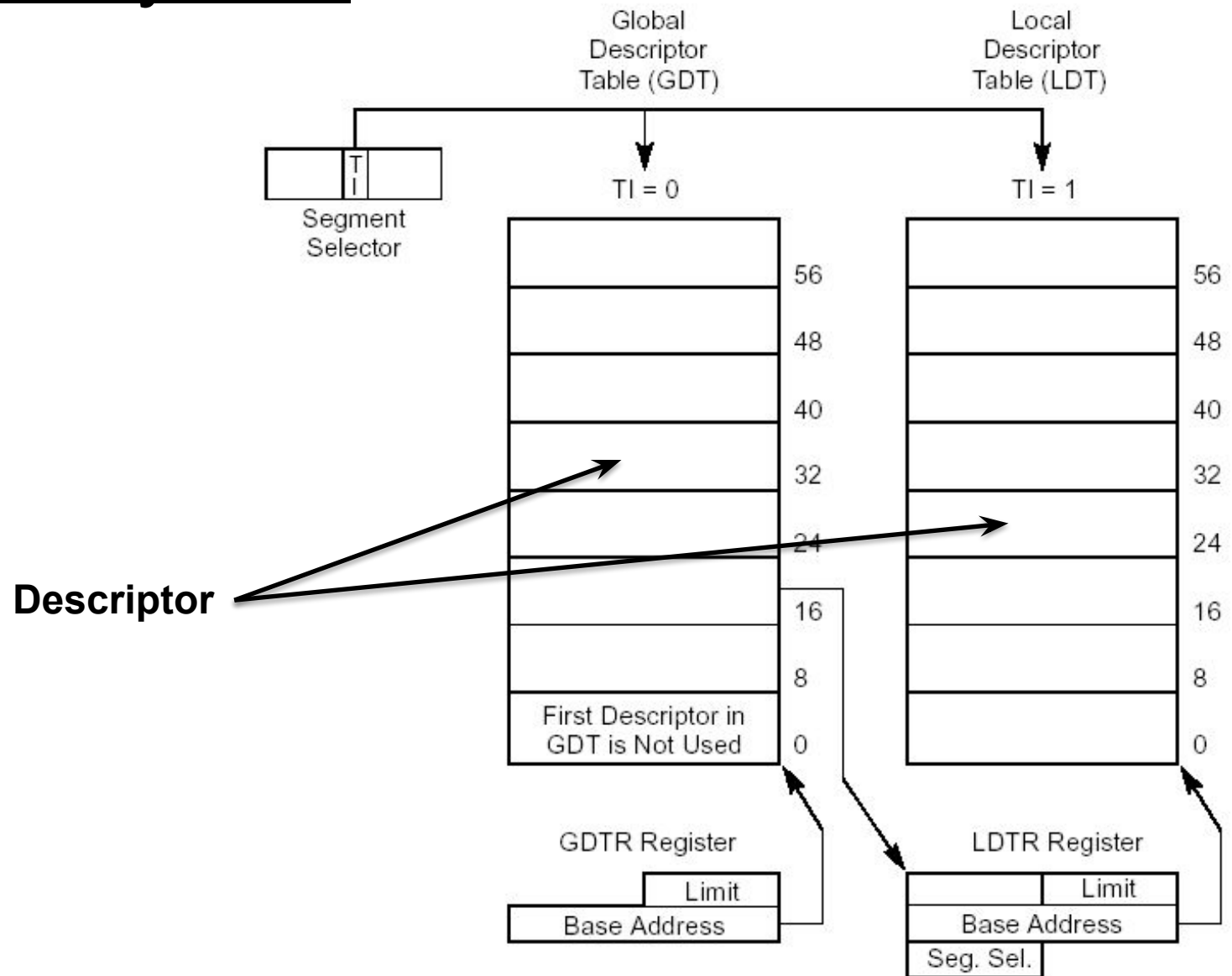


**Ejemplo: DS = 20h y Offset = 1000h**

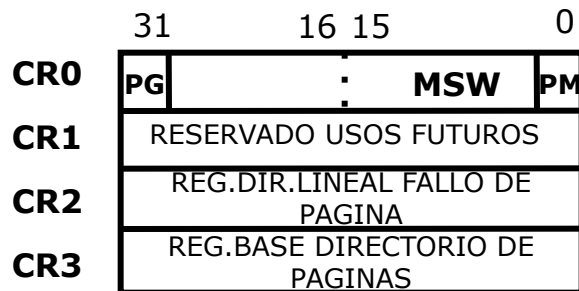
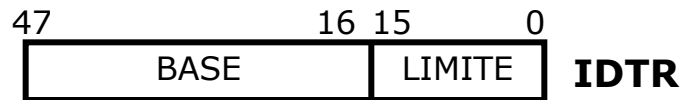
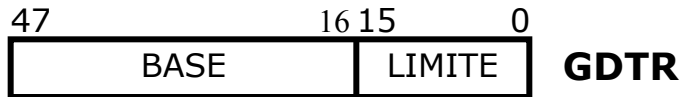
**Mov DS:[1000h], EAX**



# GDT y LDT



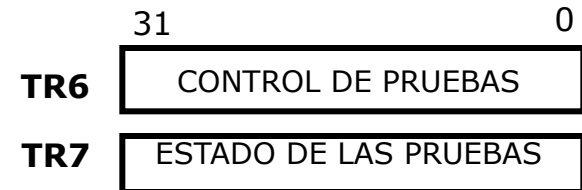
# Registros



**REGISTROS DE CONTROL**



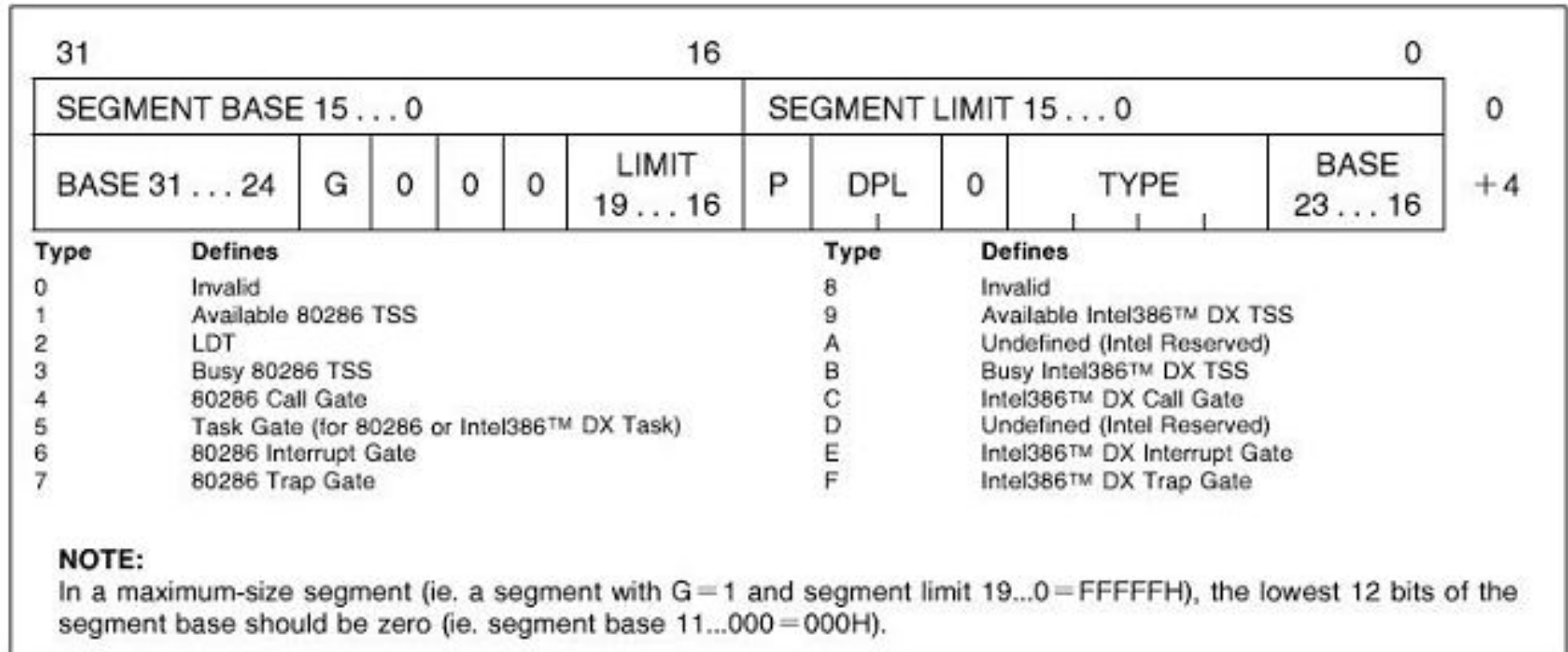
**REGISTROS DE DEBUGGING**



**REGISTROS DE PRUEBA DE LA TLB**

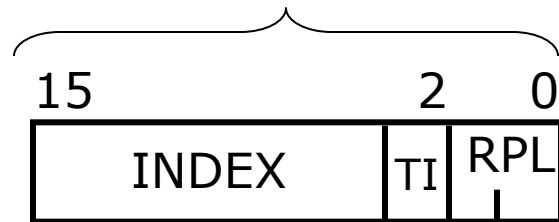
# Descriptores

Veamos la estructura genérica de los descriptores de sistemas

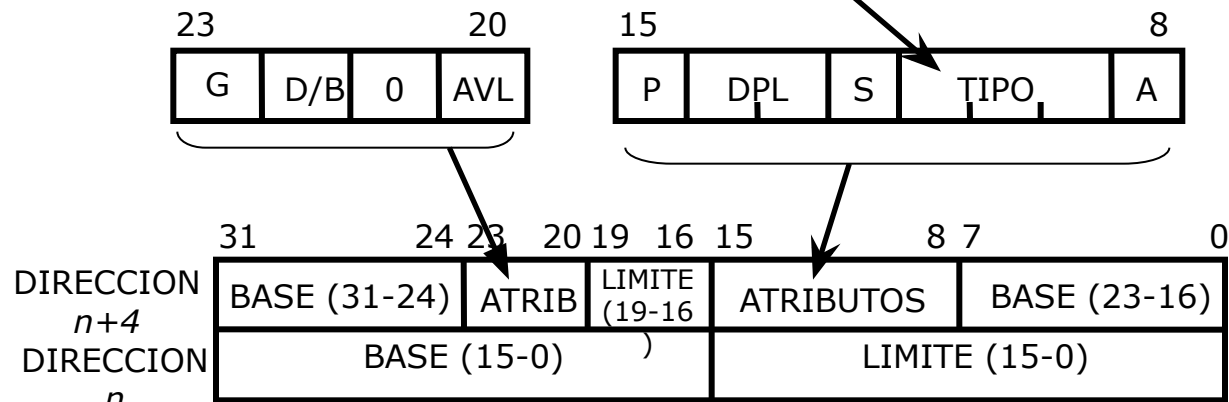
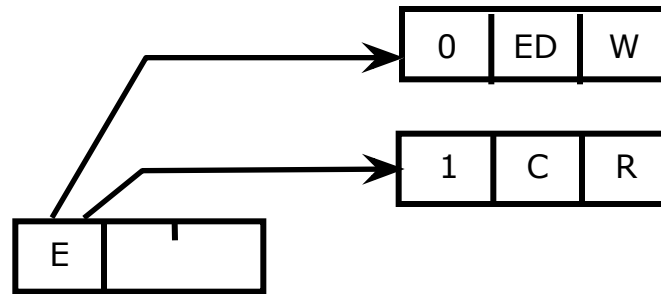


# Descriptores de segmento

SELECTOR



REGISTRO DE SEGMENTO  
(CS, DS, ES, SS, FS, o GS)



DESCRIPTOR DE SEGMENTO

# Atributos

- **P** (Bit de presencia)
  - Si  $P=1$  el segmento esta cargado en la memoria física.
  - Si  $P=0$  es segmento esta ausente.
- **DPL** (Nivel de privilegio)
  - Indica el nivel de privilegio del segmento
  - 0 = máximo privilegio, 3= minimo privilegio
- **S** (tipo de segmento)
  - Si  $S=1$  se trata de un segmento normal (código datos o pila)
  - Si  $S=0$  se trata de un segmento de sistema (puerta de llamada, TSS, etc).

# Atributos

- **Tipo:** Es un campo de tres bits.
  - En el caso de un segmento normal
    - Si E: Ejecutable es 1, se trata de un segmento ejecutable
    - El siguiente bit (C: Ajustable o conforming) define si el segmento al ser accedido cambia su nivel de privilegio al nivel de privilegio de segmento que lo llamo (C=1)
    - El ultimo bit (R: Leíble) define si el segmento de código puede ser leído (R=1).
  - Si el bit E es 0 se trata de un segmento de datos
    - El segundo bit (ED: Expansión decreciente) define si se trata de un segmento de datos normal cuando ED=0, o de un segmento de pila cuando ED=1.
    - El tercer bit (W: Escribible) define si el segmento se puede escribir cuando W=1 o si es de solo lectura cuando W=0.

# Dirección Lineal

Ejemplo: DS = 20h y Offset = 1000h

Mov DS:[1000h], EAX

DS= 0020h = 0000 0000 0010 0000 b

- Accede al Descriptor nro 4 de la GDT.
- Obtiene dirección base.
- (ej) Dirección base= 5600 0000h
- Luego suma offset de 1000 h
- Dirección lineal 5600 1000h

# Descriptores

Cuando el bit  $S=0$  se lo denomina “descriptor de sistema”.

Existen diferentes tipos, los mas usuales son:

- Trap Gate ( Puerta de excepción )
- Interrupt Gate ( Puerta de interrupción )
- Task Gate

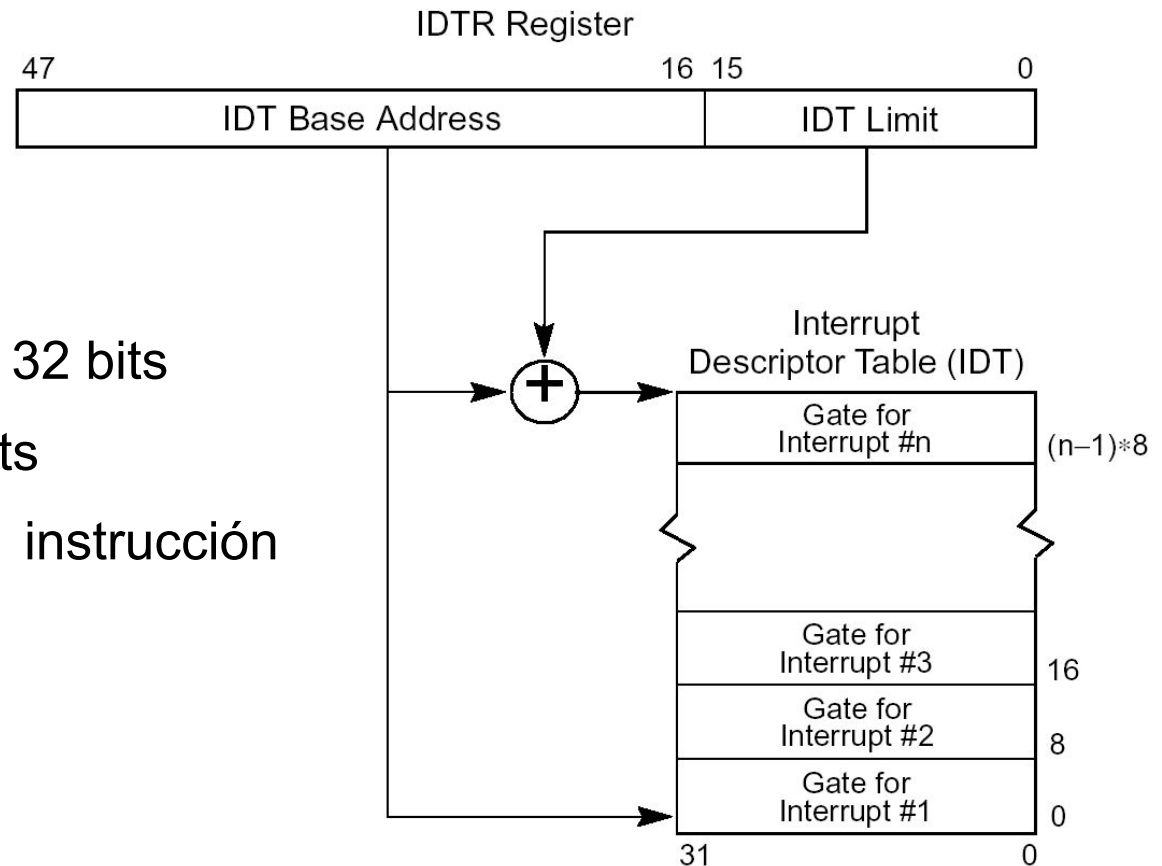




# Carga de IDT

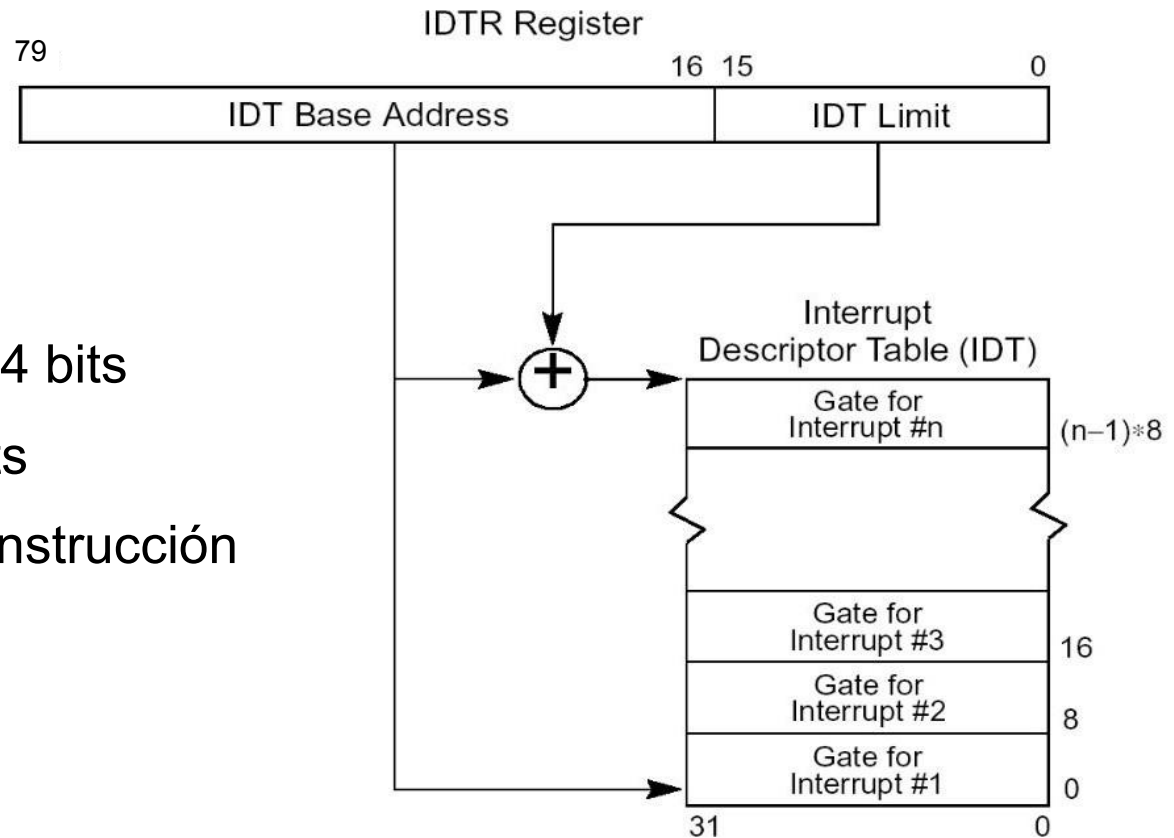
# Interrupciones en Modo protegido (32 bits)

- IDT Base Address de 32 bits
- Descriptores de 64 bits
- IDTR se carga con la instrucción LIDT

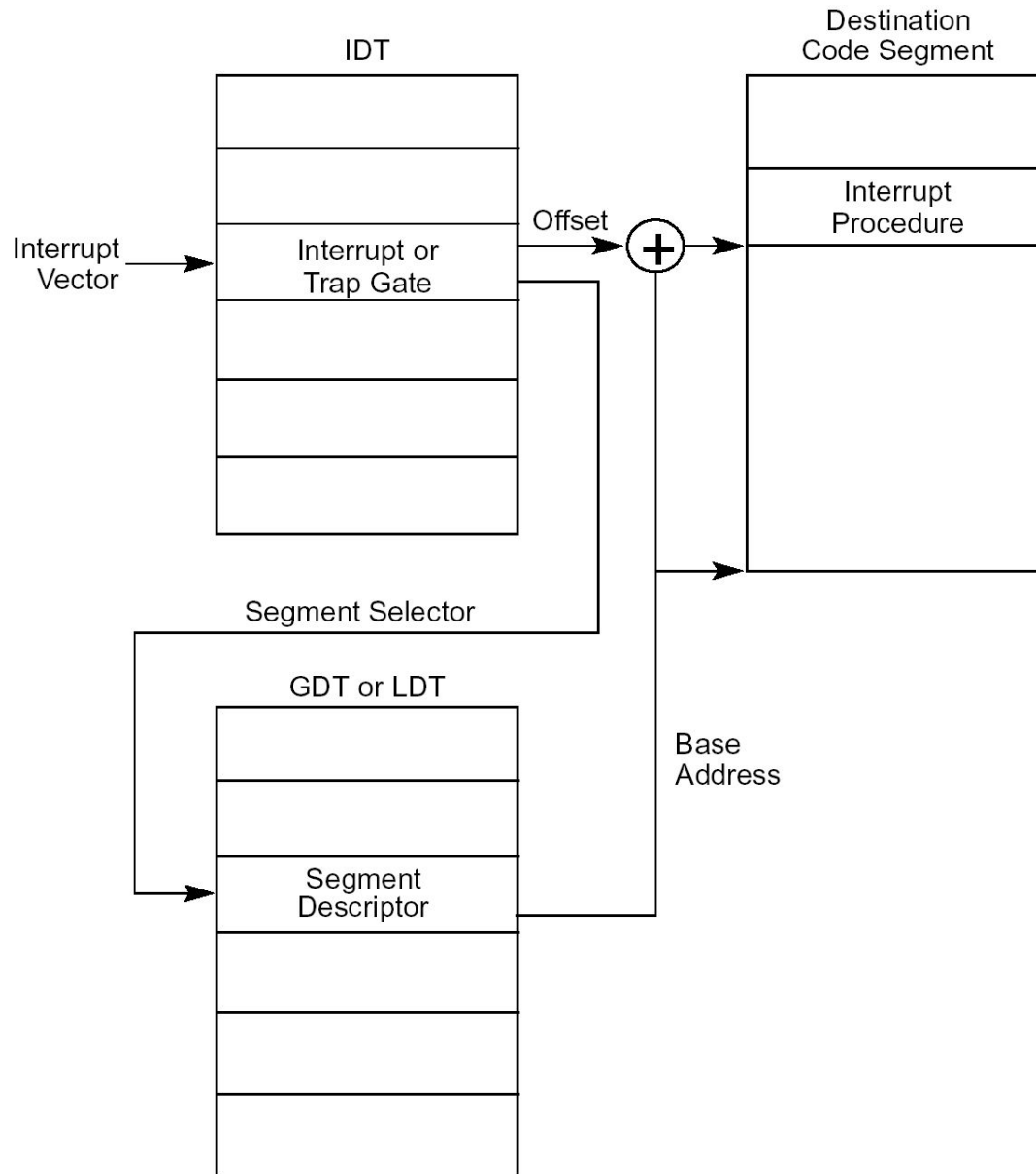


# Interrupciones en Modo protegido (64 bits)

- IDT Base Address de 64 bits
- Descriptores de 128 bits
- IDTR se carga con la instrucción LIDT



# Interrupciones en Modo protegido



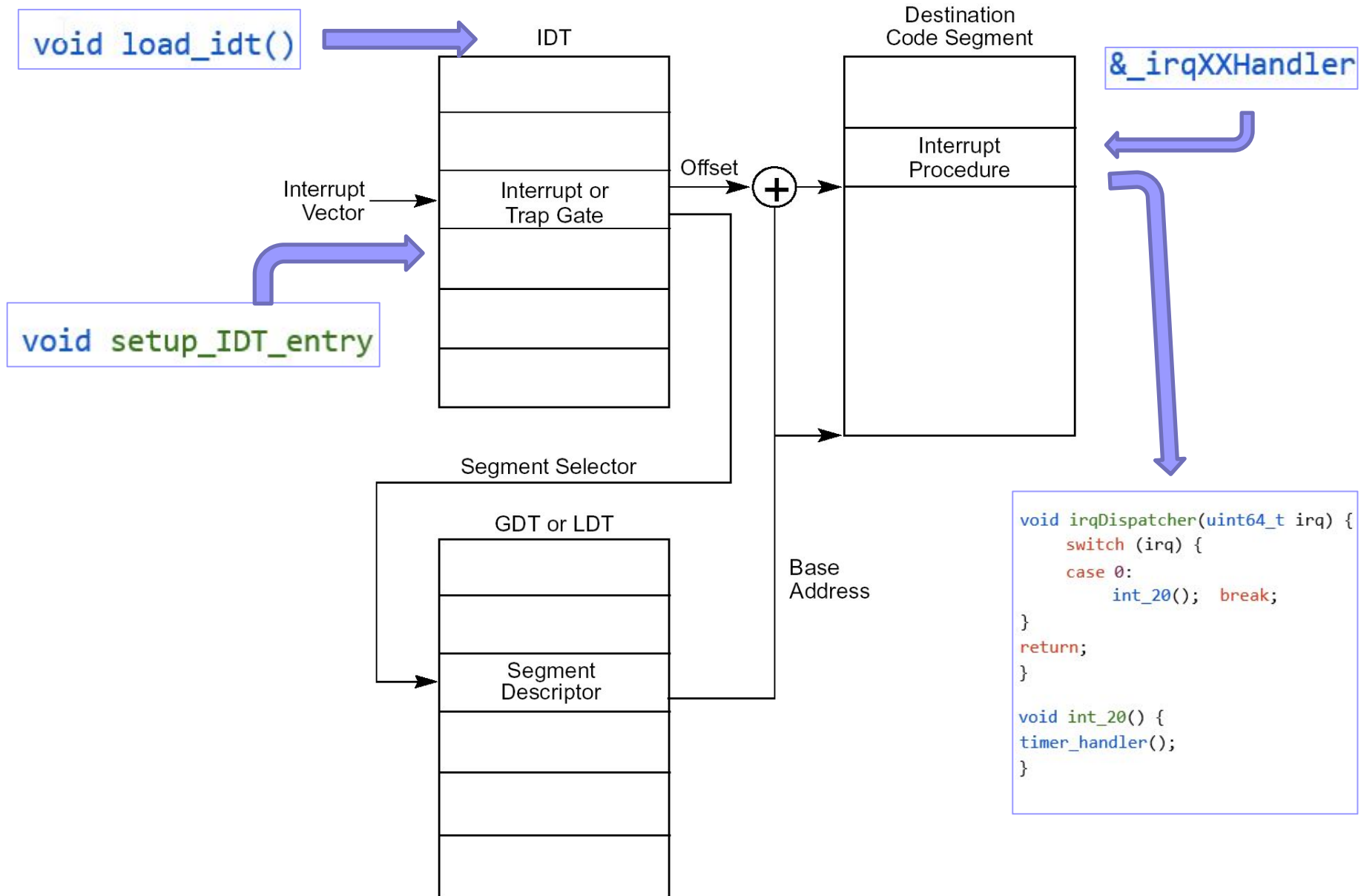
# Interrupciones en PRE\_TPE

El código necesario para cargar su propia interrupción sólo debería involucrar:

1. Definir la rutina de atención de interrupción
2. Cargar la interrupción dentro de *Load\_idt* invocando a *setup\_IDT\_entry* pasandole el número de la interrupción y el puntero a la rutina de atención de interrupción
3. Si la interrupción es de hardware, dicha rutina debe enviar el EOI y deben habilitar el IRQ con la máscara correspondiente

Pueden hallar información más detallada en [http://wiki.osdev.org/Interrupt\\_Descriptor\\_Table](http://wiki.osdev.org/Interrupt_Descriptor_Table)

# Interrupciones en Modo protegido

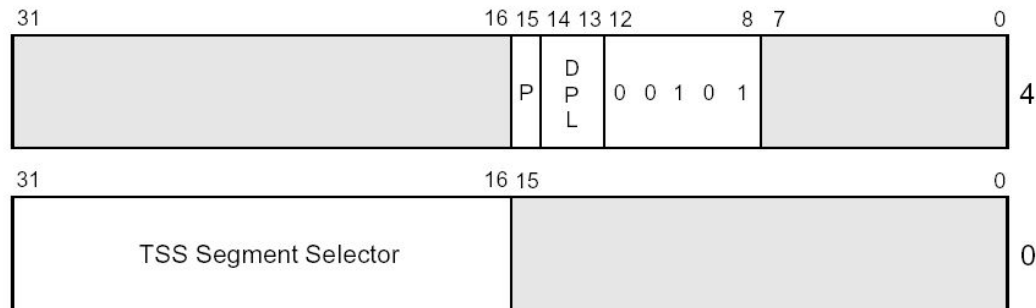


# Tabla de excepciones

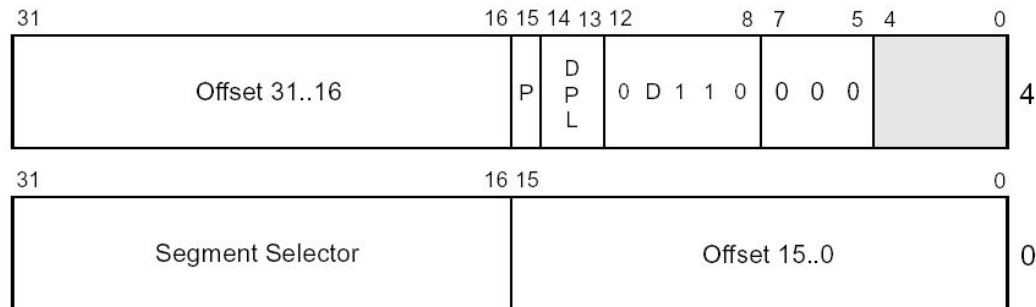
Vector No.	Mnemonic	Description	Type	Error Code	Source
0	#DE	Divide Error	Fault	No	DIV and IDIV instructions.
1	#DB	Debug	Fault/Trap	No	Any code or data reference or the INT 1 instruction.
2	—	NMI Interrupt	Interrupt	No	Nonmaskable external interrupt.
3	#BP	Breakpoint	Trap	No	INT 3 instruction.
4	#OF	Overflow	Trap	No	INTO instruction.
5	#BR	BOUND Range Exceeded	Fault	No	BOUND instruction.
6	#UD	Invalid Opcode (Undefined Opcode)	Fault	No	UD2 instruction or reserved opcode. <sup>1</sup>
7	#NM	Device Not Available (No Math Coprocessor)	Fault	No	Floating-point or WAIT/FWAIT instruction.
8	#DF	Double Fault	Abort	Yes (Zero)	Any instruction that can generate an exception, an NMI, or an INTR.
9	—	Coprocessor Segment Overrun (reserved)	Fault	No	Floating-point instruction. <sup>2</sup>
10	#TS	Invalid TSS	Fault	Yes	Task switch or TSS access.
11	#NP	Segment Not Present	Fault	Yes	Loading segment registers or accessing system segments.
12	#SS	Stack-Segment Fault	Fault	Yes	Stack operations and SS register loads.
13	#GP	General Protection	Fault	Yes	Any memory reference and other protection checks.
14	#PF	Page Fault	Fault	Yes	Any memory reference.
15	—	(Intel reserved. Do not use.)	—	No	—
16	#MF	x87 FPU Floating-Point Error (Math Fault)	Fault	No	x87 FPU floating-point or WAIT/FWAIT instruction.
17	#AC	Alignment Check	Fault	Yes (Zero)	Any data reference in memory. <sup>3</sup>
18	#MC	Machine Check	Abort	No	Error codes (if any) and source are model dependent. <sup>4</sup>
19	#XF	SIMD Floating-Point Exception	Fault	No	SSE and SSE2 floating-point instructions <sup>5</sup>
20-31	—	Intel reserved. Do not use.	—	—	—
32-255	—	User Defined (Non-reserved) Interrupts	Interrupt	—	External interrupt or INT <i>n</i> instruction.

# Tipos de descriptores en la IDT

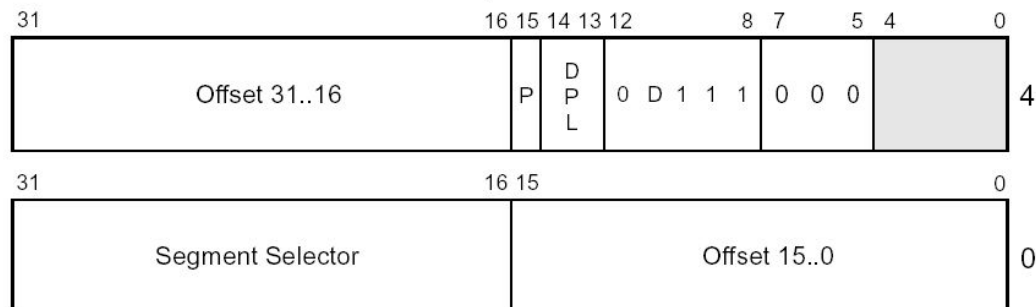
## Task Gate



## Interrupt Gate



## Trap Gate





# Manejo de tareas

**El procesador de Intel, nos da una herramienta para guardar el contexto de ejecución. Se lo denomina TSS ( Task State Segment).**

Tiene un tamaño mínimo de 100 bytes

31	15	0	
I/O Map Base Address		T	100
		LDT Segment Selector	96
		GS	92
		FS	88
		DS	84
		SS	80
		CS	76
		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
		SS2	24
ESP2			20
		SS1	16
ESP1			12
		SS0	8
ESP0			4
		Previous Task Link	0

 Reserved bits. Set to 0.

# Privilegios de E/S

**El procesador provee 2 mecanismos:**

- **Campo de 2 bits IOPL en los EFLAGS. Que especifica el mínimo nivel de privilegio que se debe tener para poder usar instrucciones de Entrada/Salida.**
- **Mapa de bits de E/S en el TSS.**

**Las instrucciones “sensibles” son:**

- **IN ( Input )**
- **OUT ( Output )**
- **INS( Input String )**
- **OUTS ( Output String )**
- **Cli ( Clear Interrupt )**
- **Sti ( Set Interrupt )**

**Para ejecutarlas el nivel de privilegio debe ser igual o menor que IOPL.**