

# Arquitectura de Computadores II

## Clase 15

Introducción al modo protegido

Facultad de Ingeniería  
Universidad de la República  
Instituto de Computación

# Contenido

- Generalidades.
- Modos de operación.
- Manejo de traps.

# Intel 32 bits

- Breve reseña histórica:
  - 1982 Intel introduce el 286 de 16 bits
    - Modo protegido, manejo de memoria virtual
  - 1985 aparece el 386, primer procesador Intel de 32 bits
    - Direcccionamiento de 32 bits (4Gb de memoria física)
    - Memoria segmentada
    - Paginación
  - 1989 introduce el 486
    - Cache de nivel 1 (on-chip cache)
    - On-chip 387 math coprocessor (integrado en el CPU)
    - Manejo de ahorro del consumo
  - 1993 introduce procesadores Pentium
    - Segundo pipe de ejecución
    - Duplica la cache
    - Protocolo MESI para mejorar manejo de la cache
    - Branch prediction
    - APIC (Advanced Programmable Interrupt Controller), soporte para múltiples procesadores, manejo interrupciones internas (timer interno) y externas
    - Tecnología MMX

# Modo Protegido Arquitectura Intel

- Breve introducción a las nuevas funcionalidades ofrecidas por la arquitectura Intel 386 en modo protegido
  - Espacio de memoria extendido y protegido
  - Funcionalidades destinadas a facilitar un ambiente multi-tarea
  - Paginación
  - Modo 8086 virtual

# Modo Protegido Arquitectura Intel

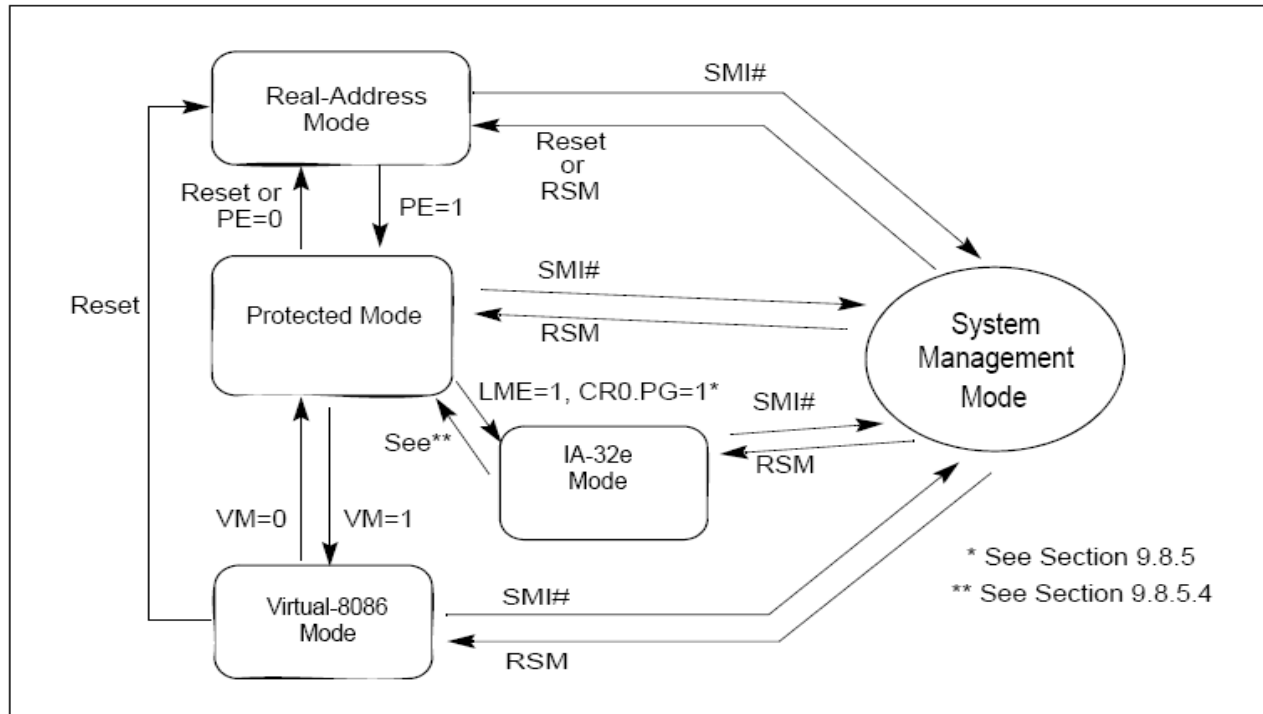
- Brinda múltiples funcionalidades que potencian la multitarea y mejoran la estabilidad del sistema
- Entre estas funcionalidades está la protección de memoria, paginación y soporte de un manejo de memoria virtual, a través de la MMU (memory management unit)
- La mayoría de los S.O. actuales como Windows y Linux corren en modo protegido
- Modo Real deshabilita estas mejoras para brindar compatibilidad hacia atrás (DOS)

# Modos de Operación

- Modo Real (procesador arranca en este modo)
- Modo Protegido
- System Management Mode (SMM)
  - Manejo de energía, mediante una interrupción externa se accede a este modo y el procesador brinda un espacio de memoria separado preservando contexto
- Modo Virtual-8086
  - Habilita correr software 8086 (16 bits) en un entorno protegido
- Modo IA-32
  - En arquitecturas de 64 bits, permite compatibilidad con software 32bits

# Modos de Operación

Figure 2-3 shows how the processor moves between operating modes.



**Figure 2-3. Transitions Among the Processor's Operating Modes**

The processor is placed in real-address mode following power-up or a reset. The PE flag in control register CR0 then controls whether the processor is operating in real-address or protected mode. See also: Section 9.9, "Mode Switching."

# Protección

- Existen cuatros niveles de privilegio
  - Numerados del 0 al 4 (a mayor número menores privilegios)
  - Restringen acceso a memoria de datos, código, etc
  - Restringen las instrucciones accesibles

PROTECTION

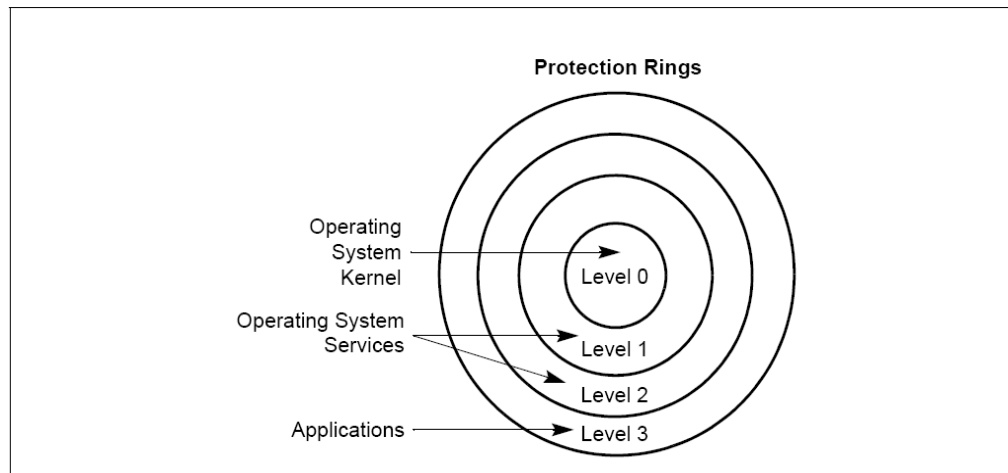


Figure 4-3. Protection Rings



# Registros de Control

- Existen cinco registros de 32 bits, CR0 .. CR4, que determinan:
  - el modo de funcionamiento del procesado
  - características de la actual tarea en ejecución
- CR0 controla el modo de operación y el actual estado del procesador
- CR1 reservado
- CR2 y CR3 usado por el sistema de paginación de memoria
- CR4 habilita diversas extensiones de la arquitectura
- Todos estos registros sólo pueden ser manipulados en el nivel 0 de privilegio

# Registros de Control

- Algunas banderas del CR0
  - **PG habilita la paginación de la memoria**
  - CD habilita la cache
  - NW configura la cache
  - AM habilita el chequeo automático de alineamiento a 16 bits.
  - WP habilita la escritura de páginas read only por rutinas con mayores privilegios
  - NE numeric error FPU (Floating-point unit)
  - ET en 1 en procesadores Pentium, en 386 y 486 indica el soporte de instrucciones Intel 387 DX coprocesador matemático
  - **PE habilita el modo protegido (bit 0)**

# Registro de Banderas

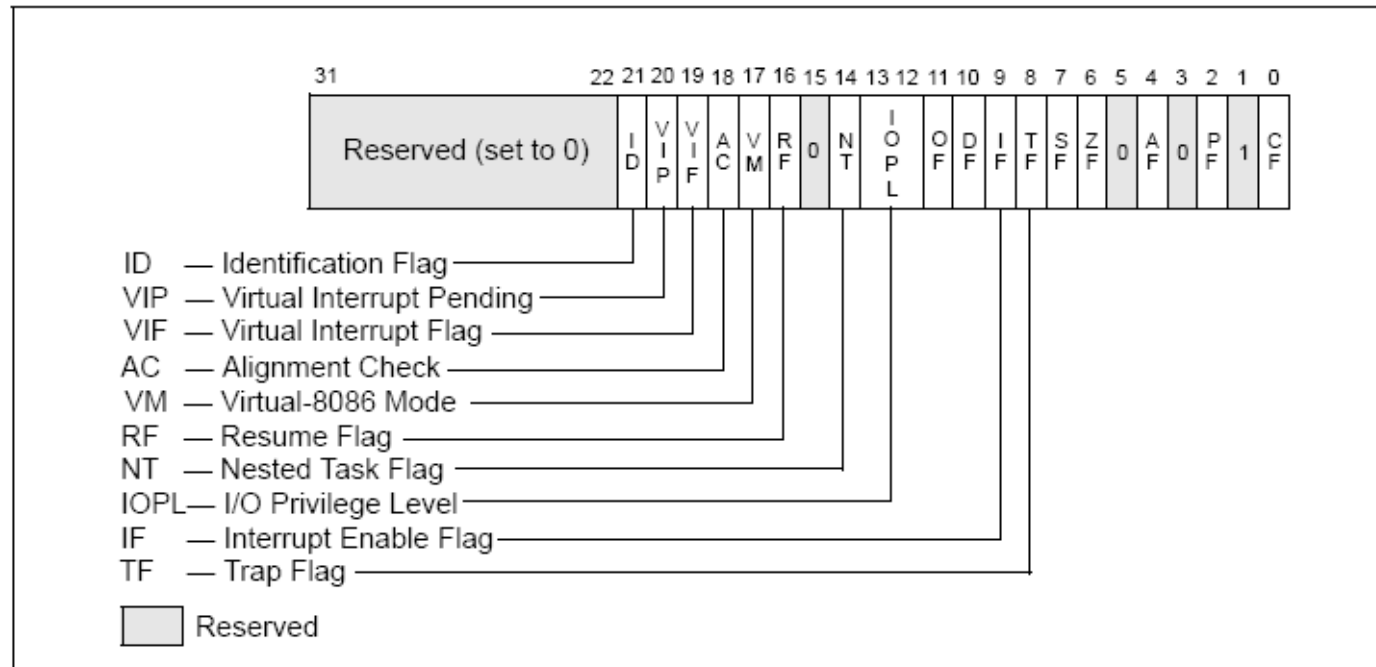


Figure 2-4. System Flags in the EFLAGS Register

- IF controlada por cli, sti, iret
- TF procesador genera una excepción de debug para permitir correr paso a paso
- IOPL contiene el actual nivel de privilegio

# Direccionamiento Segmentado de Memoria (MMU)

- Independientemente del modo se mantiene el concepto de segmentos, a partir de los cuales se obtiene la dirección base de memoria donde comienza cada segmento, y al que se le suma un determinado offset
- Existen diversos registros que identifican los segmentos: CS, DS, SS, ES, FS, GS
- Cada uno de ellos refiere a determinado tipo de segmento: code, data, stack.
- Facilitan la separación lógica de la información de distintas tareas, de forma que puedan correr en el mismo procesador sin interferir unas con otras.

# Direccionamiento Segmentado de Memoria

- Modo Real, el registro de segmento contiene los 16 bits del alto orden de una dirección base lineal de 20 bits
- Modo Protegido, el registro de segmento es un índice en una tabla de estructuras de datos que contiene diversas propiedades de cada segmento
  - Dirección base
  - Límite
  - Nivel de privilegio
  - Otras propiedades

# Direcccionamiento Segmentado de Memoria

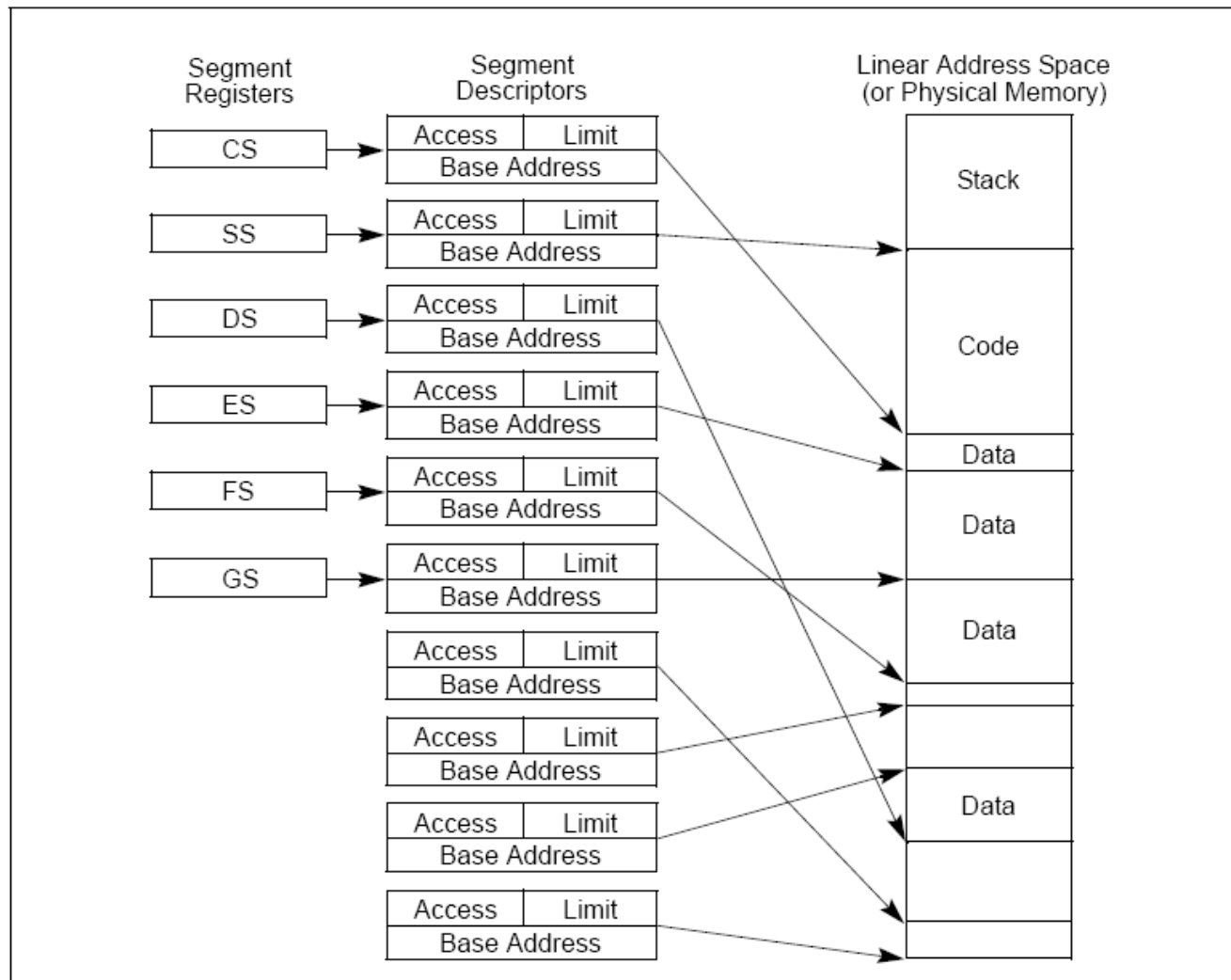


Figure 3-4. Multi-Segment Model

# Direccionamiento Segmentado de Memoria

- Memoria física  $2^{32}$  bytes = 4 GigaBytes
  - Pudiendo ser mapeada en memoria de lectura-escritura, solo lectura, memoria I/O
- Existen dos nivel de transformaciones para llevar de una dirección lógica a una física:
  - Logical address translation
  - Linear address space paging (opcional)

# Logical to linear address translation

## PROTECTED-MODE MEMORY MANAGEMENT

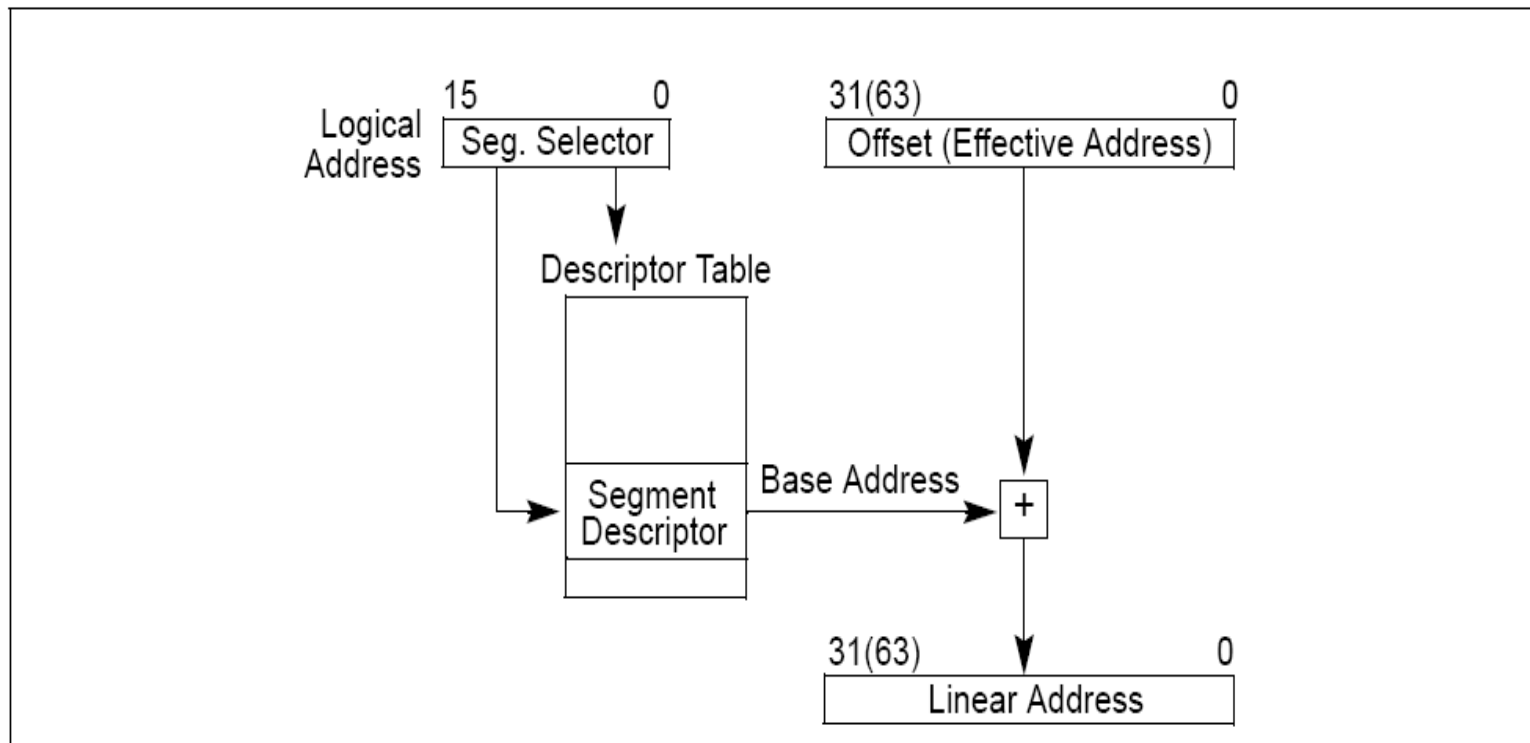


Figure 3-5. Logical Address to Linear Address Translation



# Paging (demand paged virtual memory)

- En modo protegido esta arquitectura permite que el espacio de memoria lineal de 4GBytes sea mapeado directamente en una memoria física del mismo tamaño o indirectamente mediante el paginado en una memoria física de menor tamaño
- Páginas de largo fijo (4KBytes, 2MBytes o 4MBytes) que son mapeadas en memoria o disco.
- Si en el momento de acceder a una página ésta no se encuentra en memoria, se produce una excepción, permitiendo el manejo de un sistema de memoria virtual.
- También se puede lograr con los segmentos.

# Paging (virtual memory)

## Linear to physical address translation

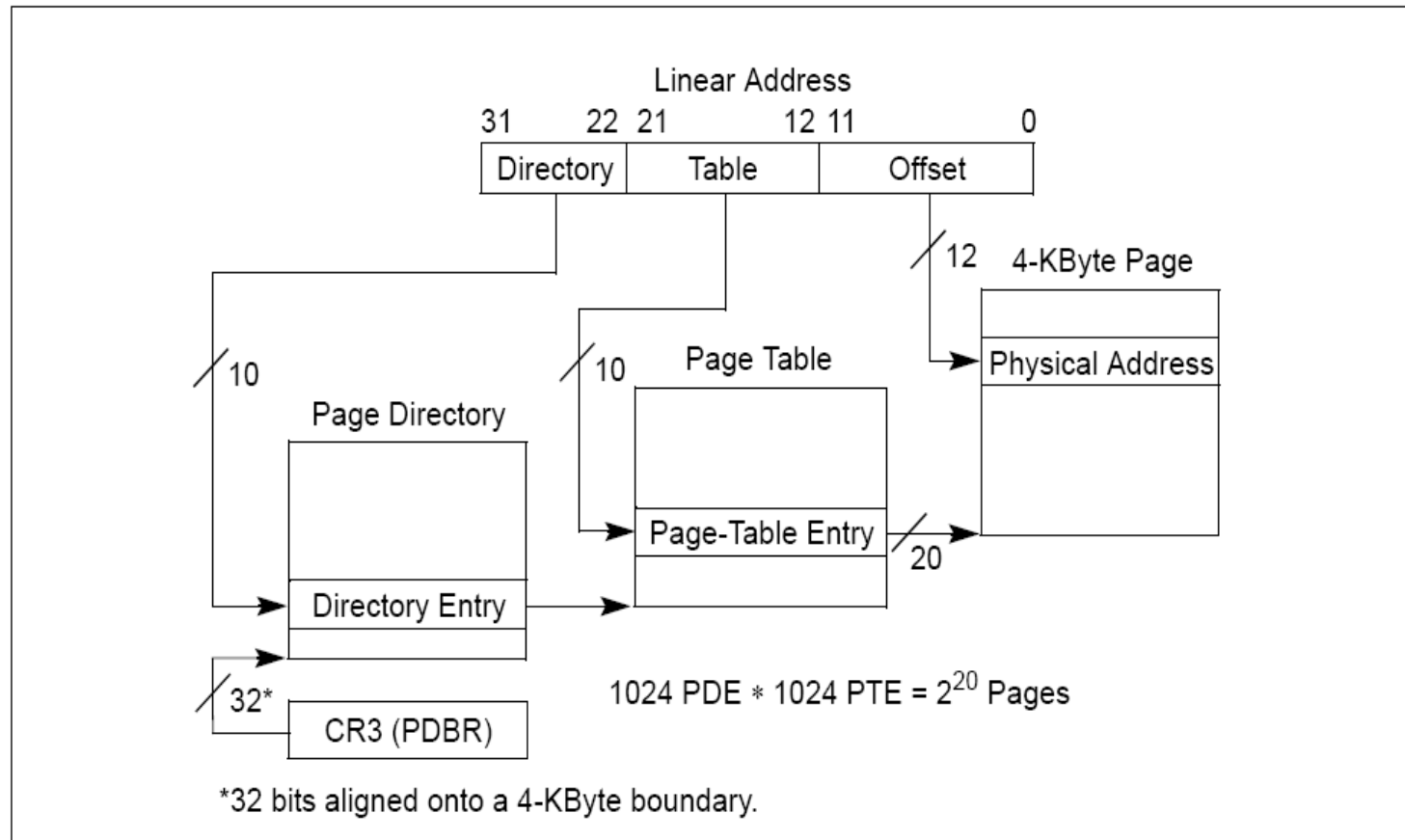


Figure 3-12. Linear Address Translation (4-KByte Pages)

# Direccionamiento de Memoria

## Segmentación + Paginación

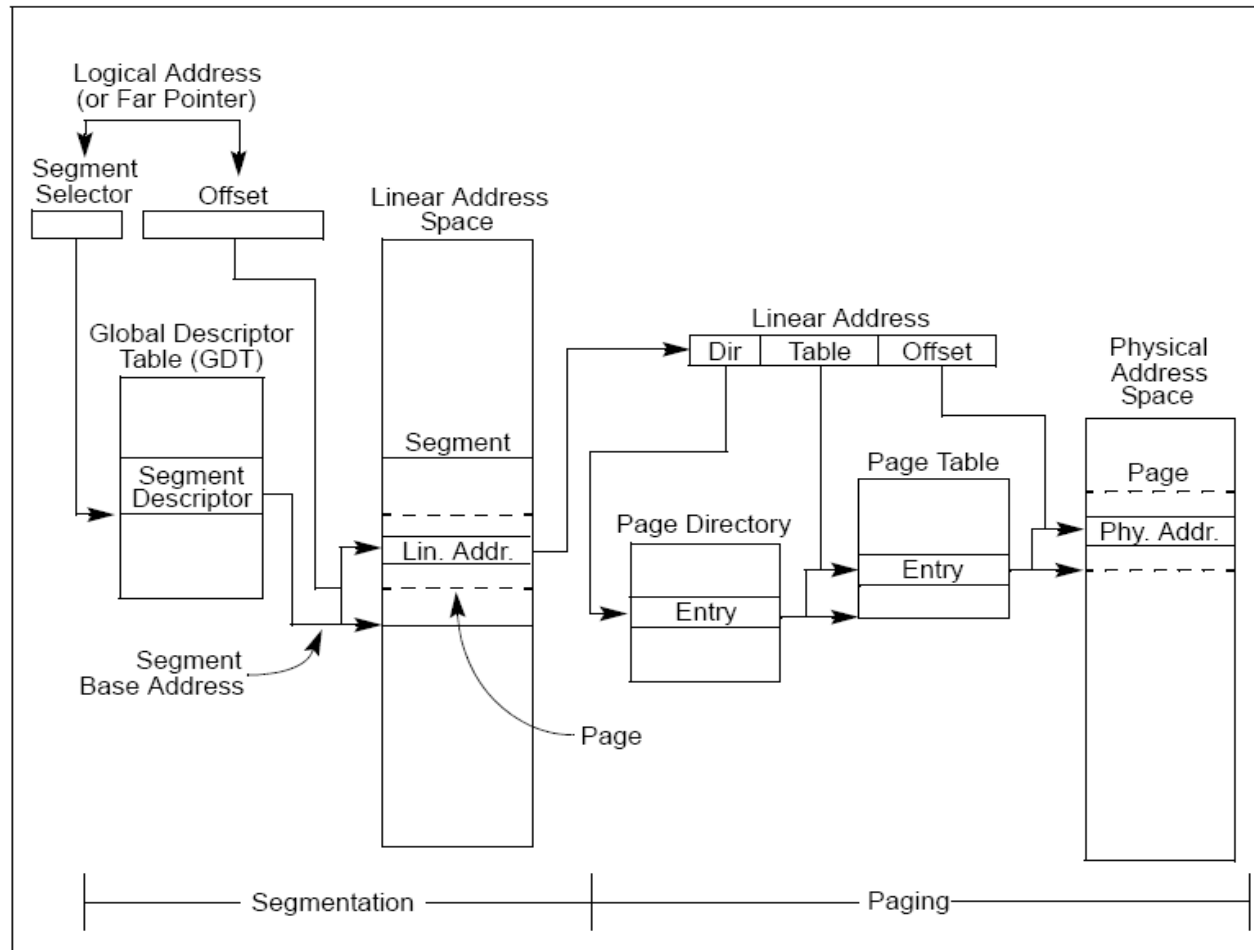
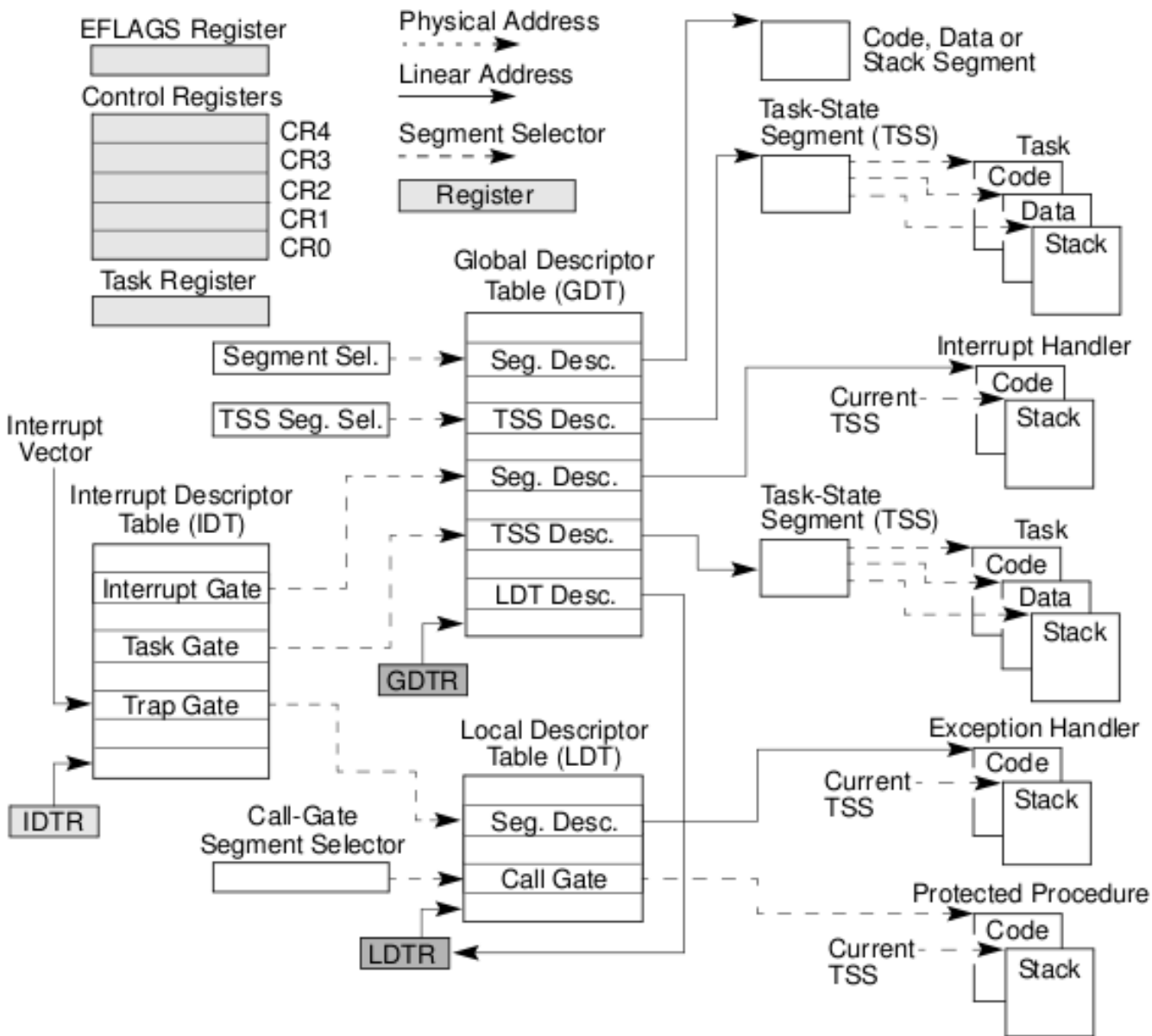


Figure 3-1. Segmentation and Paging

# Registros y estructuras de datos



# Registros de Organización de Memoria

- System Table Registers  
Contienen direcciones base y límites de las tablas que controlan el manejo de memoria e interrupciones
  - GDTR Global Descriptor Table Register
  - IDTR Interrupt Descriptor Table Register
- System Segment Register  
Contienen posiciones en la GDT
  - LDTR Local Descriptor Table Register
  - TR Task Register
- Las direcciones base de estas tablas tienen que estar alineadas a fronteras de 8 bytes

# Segment Descriptor Tables

- Son tablas de segment descriptors, existen dos tipos: GDT y LDT

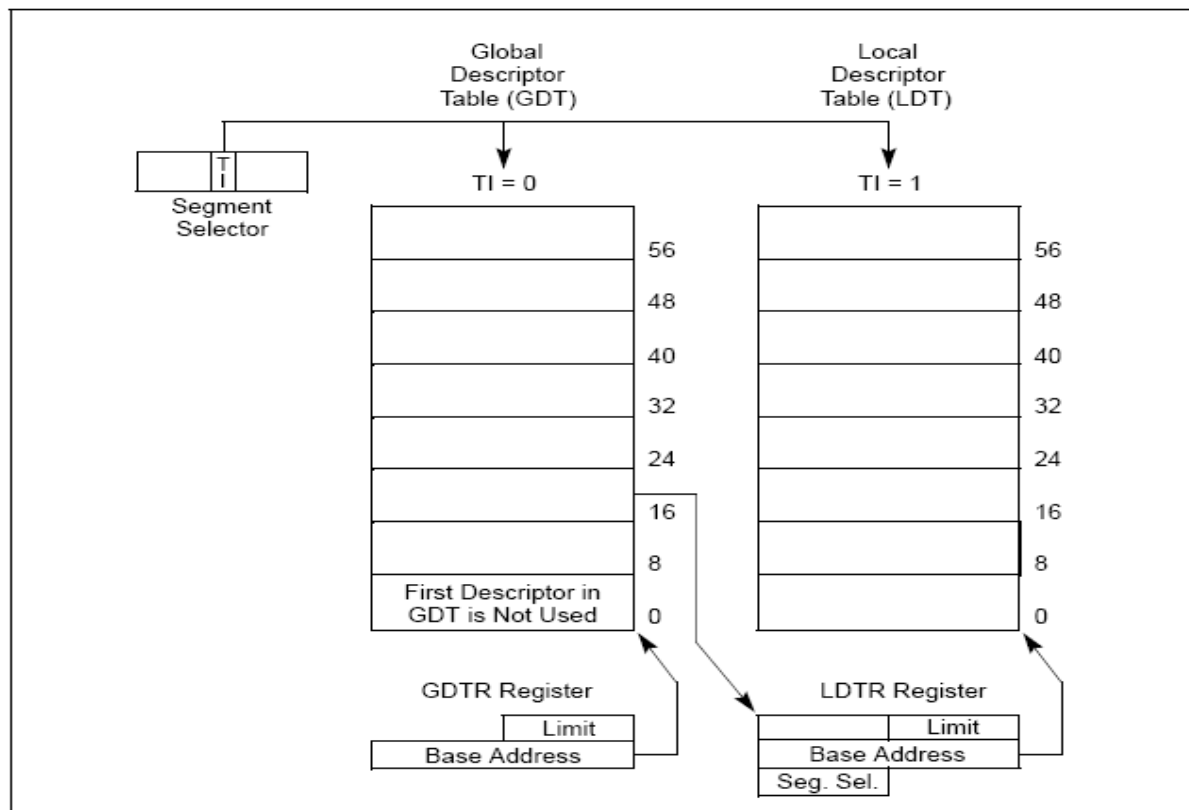


Figure 3-10. Global and Local Descriptor Tables

# Segment register

- Existen seis registros de segmento de 16 bits: CS, DS, SS, ES, FS, GS

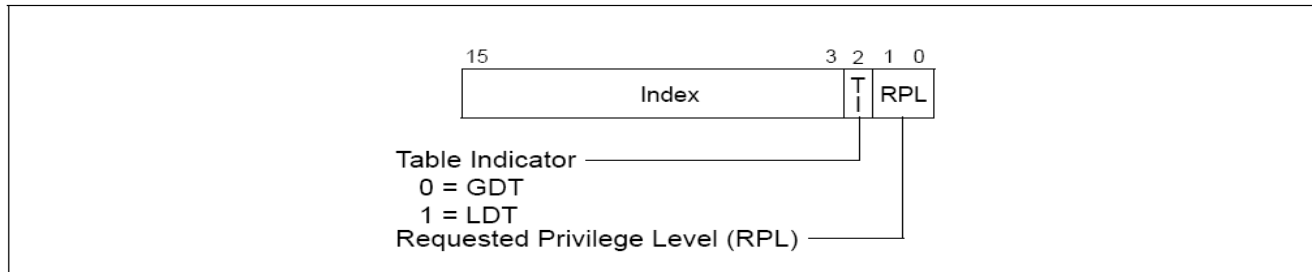


Figure 3-6. Segment Selector

Visible Part		Hidden Part	
Segment Selector		Base Address, Limit, Access Information	
			CS
			SS
			DS
			ES
			FS
			GS

Figure 3-7. Segment Registers

# Tipos de segmentos

Table 3-1. Code- and Data-Segment Types

Type Field					Descriptor Type	Description
Decimal	11	10 E	9 W	8 A		
0	0	0	0	0	Data	Read-Only
1	0	0	0	1	Data	Read-Only, accessed
2	0	0	1	0	Data	Read/Write
3	0	0	1	1	Data	Read/Write, accessed
4	0	1	0	0	Data	Read-Only, expand-down
5	0	1	0	1	Data	Read-Only, expand-down, accessed
6	0	1	1	0	Data	Read/Write, expand-down
7	0	1	1	1	Data	Read/Write, expand-down, accessed
		<b>C</b>	<b>R</b>	<b>A</b>		
8	1	0	0	0	Code	Execute-Only
9	1	0	0	1	Code	Execute-Only, accessed
10	1	0	1	0	Code	Execute/Read
11	1	0	1	1	Code	Execute/Read, accessed
12	1	1	0	0	Code	Execute-Only, conforming
13	1	1	0	1	Code	Execute-Only, conforming, accessed
14	1	1	1	0	Code	Execute/Read-Only, conforming
15	1	1	1	1	Code	Execute/Read-Only, conforming, accessed



# Interrupciones y Excepciones

- Interrupciones
  - Ocurren asincrónicamente por eventos de hardware y también pueden ser invocadas por software
- Excepciones
  - Procesador detecta condiciones de error al ejecutar una instrucción (división por cero), violaciones de protección, páginas faltantes, etc
  - Se dividen en:
    - Faults: excepción que puede ser corregida y volver luego al flujo normal de ejecución (páginas faltantes). Se retorna a la instrucción que provocó la excepción
    - Traps: excepción reportada luego de ejecutar una instrucción, se vuelve a la instrucción siguiente
    - Aborts: excepción que no reporta precisamente el lugar que provocó la propia excepción, no permiten retomar el flujo normal (errores de hardware, valores ilegales en las tablas del sistema)

# Interrupciones y Excepciones

- IDT Interrupt Description Table
  - Contiene un arreglo de gate descriptors, los cuales proveen acceso a las rutinas de atención de interrupciones
  - Se carga y recupera mediante las instrucciones protegidas de sidt y lidt
  - Los gate descriptors en esta tabla pueden ser de los siguientes tipos: interrupt, trap o task gate descriptors

# Interrupciones y Excepciones

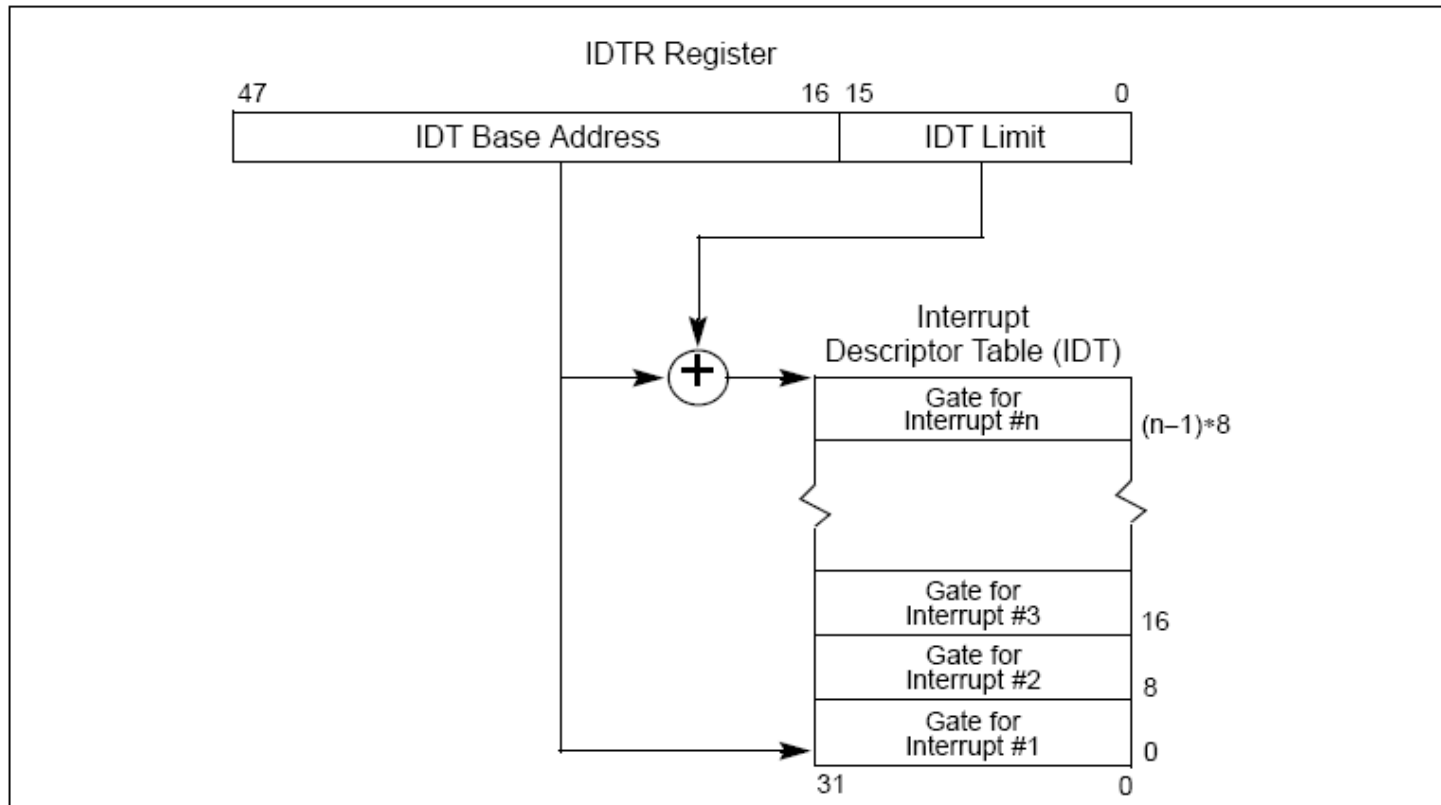


Figure 5-1. Relationship of the IDTR and IDT

# Call, interrupt y trap gates

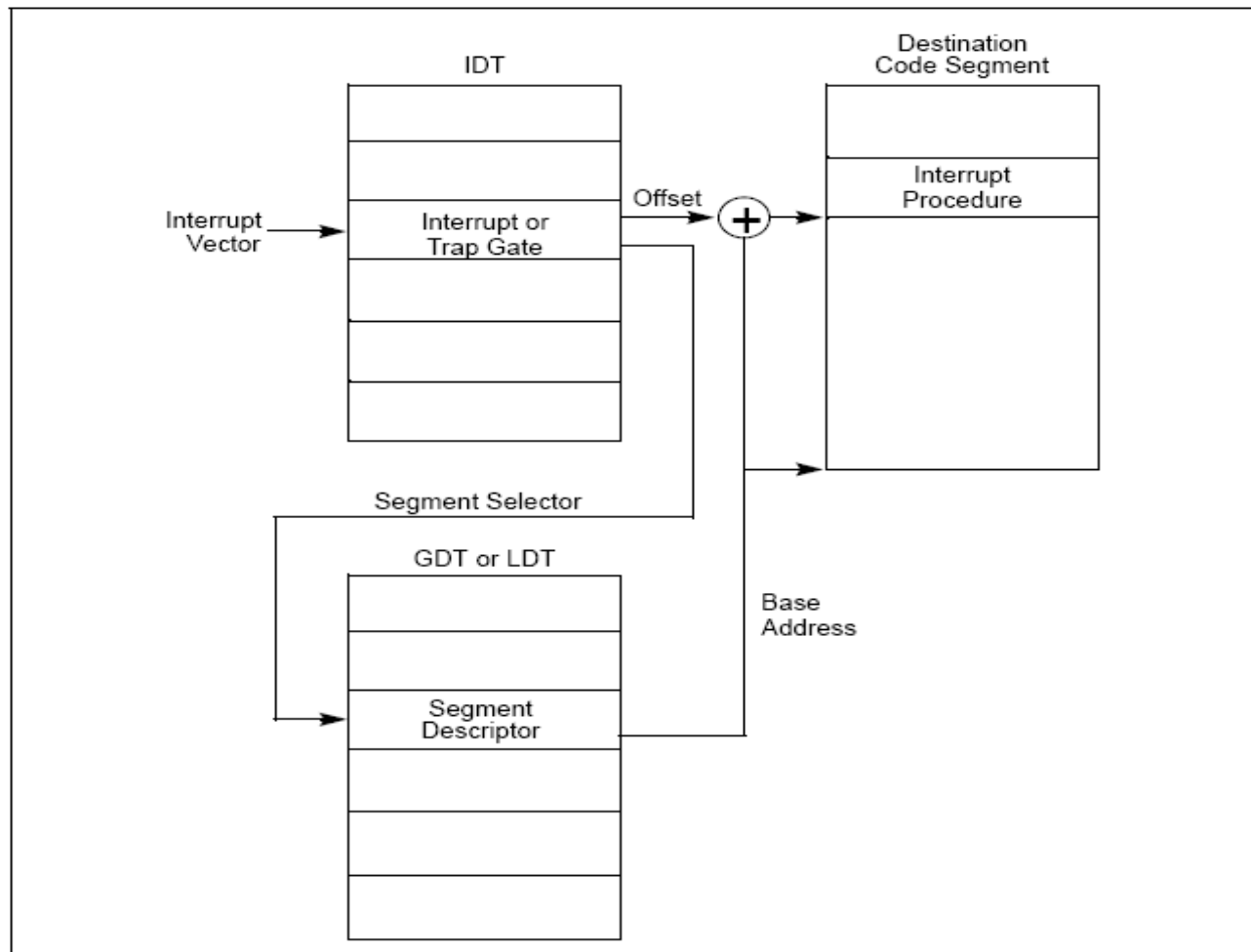


Figure 5-3. Interrupt Procedure Call

# Tasks

- Facilidades para el manejo de tareas
- Una tarea esta formada por:
  - un entorno de ejecución
    - CS, DS, SS (stacks separados para distintos niveles de protección)
  - un TSS (Task-state segment)
    - Especifica los segmentos que conforman el entorno de ejecución y provee espacio de almacenamiento para guardar el estado de una tarea.
    - En ambientes multitarea brinda una forma de anidar tareas
- Cada tarea es identificada por el segment selector que direcciona su TSS en el GDT
- Si se habilita la paginación CR3 se carga con la dirección de directorio de paginas utilizado por la tarea

# Tasks

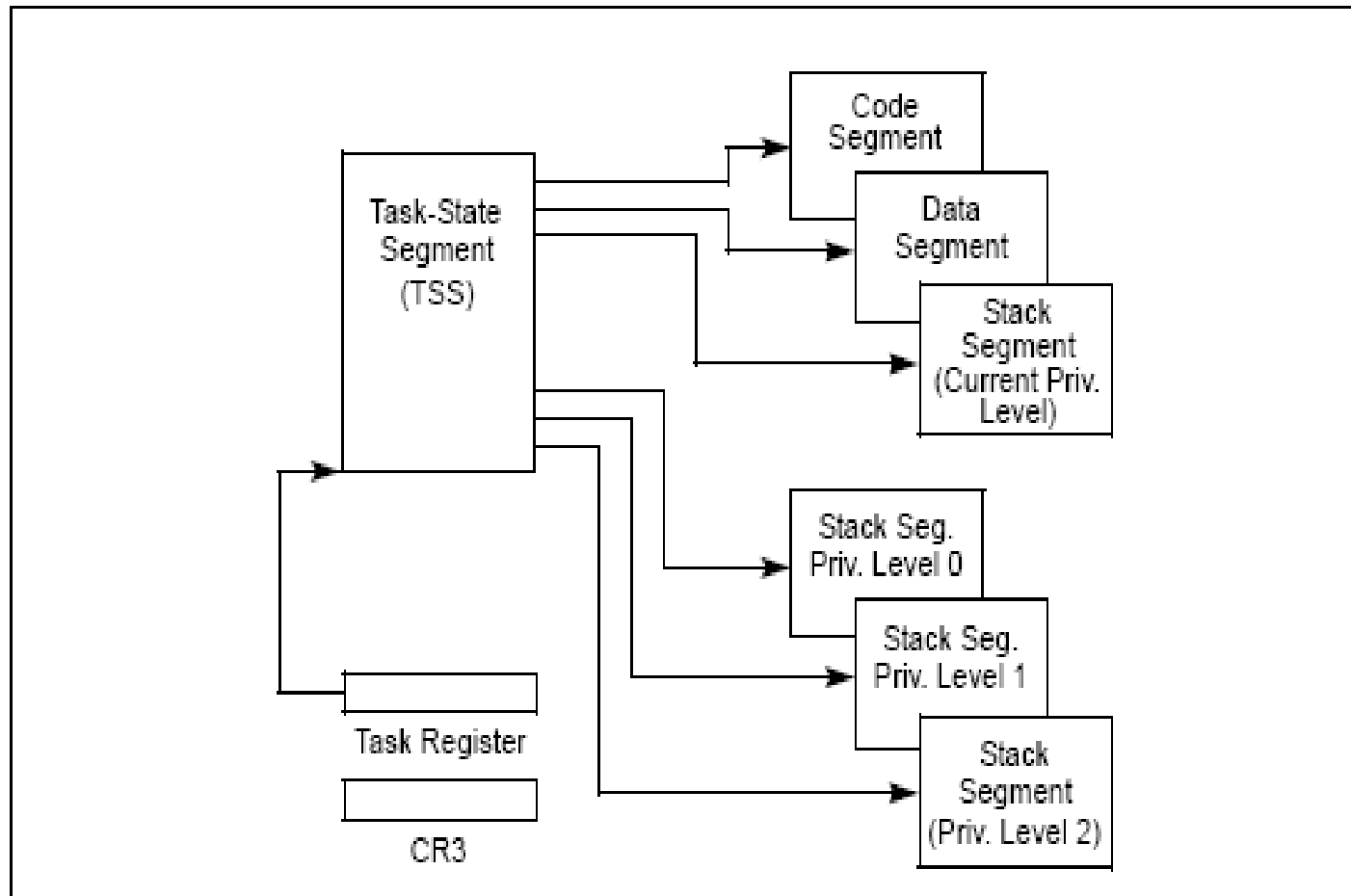



Figure 6-1. Structure of a Task

# Task-State Segment

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.

- Contexto de una tarea cargado y salvado automáticamente por hardware
- Existen 3 niveles de stack, de forma que invocaciones desde distintos niveles de protección no compartan el stack

Figure 6-2. 32-Bit Task-State Segment (TSS)

# Ejecución de una tarea

- Un software o el procesador puede despachar una tarea en una de las siguientes formas:
  - Software (explícito )
    - Call a una tarea (call a un task-gate descriptor)
    - Jump a una tarea (jmp a un task-gate descriptor)
  - Procesador (implícito)
    - Call del procesador a un interrupt-handler task
    - Call del procesador a un exception-handler task
    - Un iret cuando el NT en las flags está prendido
- Inicialmente se debe cargar en el TR (task register) mediante la instrucción ldt, el TSS de la actual tarea



# Referencias

- Intel, Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, System Programming Guide, Part 1, May 2007.

# Preguntas