X86 – Modo protegido (32 bits)

(parte 2)

Segmentos de Sistema

- Segmentos que não são:
  - Código
  - Nem Dados
  - Nem Pilha
- O que são?
  - TSS: Task-state segment
  - LDT: Local Descriptor Table
- Tecnicamente, a GDT não é considerada um segmento
  - Por não ser acessada por meio de seletor de segmento e um descritor de segmento

Descritores para segmentos de sistema/gate

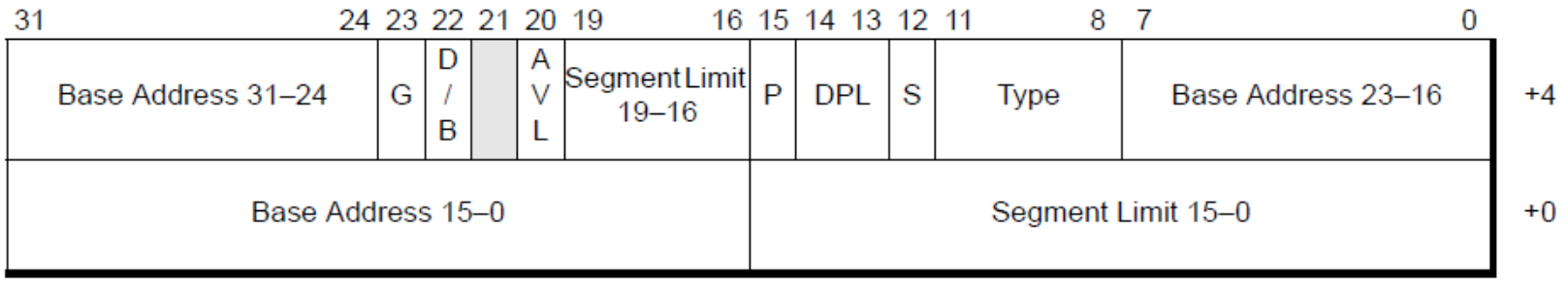

Figure 4-13. Generic Segment Descriptor—Legacy Mode

Lembrete:

Flag S (tipo de descritor):
Especifica se o descritor de segmento é para um segmento de sistema (S é 0) ou um segmento de código/dados (S é 1)

Vimos os casos para S = 1, veremos agora o que significa S = 0...

Descritores para segmentos de sistema/gate

- Quando o flag S é 0, o descritor é para um segmento de sistema.
- O processador reconhece os seguintes tipos de descritores de sistema:
    - Descritor de segmento para LDT (local descriptor-table)
    - Descritor para TSS (task-state segment)
    - Descritor para Call-gate
    - Descritor para Interrupt-gate
    - Descritor para Trap-gate
    - Descritor para Task-gate
  - Podemos dividir os segmentos de sistema então em duas categorias:
    - Descritores de segmento de sistema:
        - LDT e TSS
    - Descritores “gate”
        - Descritor para Call-gate
        - Descritor para Interrupt-gate
        - Descritor para Trap-gate
        - Descritor para Task-gate

Descritores para segmentos de sistema

Table 3-2. System-Segment and Gate-Descriptor Types

Type Field					Description	
Decimal	11	10	9	8	32-Bit Mode	IA-32e Mode
0	0	0	0	0	Reserved	Upper 8 byte of an 16-byte descriptor
1	0	0	0	1	16-bit TSS (Available)	Reserved
2	0	0	1	0	LDT	LDT
3	0	0	1	1	16-bit TSS (Busy)	Reserved
4	0	1	0	0	16-bit Call Gate	Reserved
5	0	1	0	1	Task Gate	Reserved
6	0	1	1	0	16-bit Interrupt Gate	Reserved
7	0	1	1	1	16-bit Trap Gate	Reserved
8	1	0	0	0	Reserved	Reserved
9	1	0	0	1	32-bit TSS (Available)	64-bit TSS (Available)
10	1	0	1	0	Reserved	Reserved
11	1	0	1	1	32-bit TSS (Busy)	64-bit TSS (Busy)
12	1	1	0	0	32-bit Call Gate	64-bit Call Gate
13	1	1	0	1	Reserved	Reserved
14	1	1	1	0	32-bit Interrupt Gate	64-bit Interrupt Gate
15	1	1	1	1	32-bit Trap Gate	64-bit Trap Gate

Call Gates, Trap Gates, Interrupt Gates, Task Gates

- Os descritores do tipo “Gate” proporcionam um acesso controlado a segmentos de código com nível de privilégio diferentes.
- Sem passar por Gates, o comportamento do controle de acesso de um segmento de código para outro se ocorre da seguinte maneira:
    - O segmento de código de destino é nonconforming: o segmento de código de origem tem que ter nível de acesso (CPL) igual ao nível requerido pelo segmento de destino (DPL). O RPL do seletor de segmento pode ser numericamente menor ou igual ao CPL do segmento de origem. O CPL não muda com a transferência.
    - O segmento de código de destino é conforming: o CPL do segmento de origem tem que ser numericamente igual ou maior (privilégio menor) do que o DPL do segmento de destino. O processador gera uma general-protection exception somente se o CPL é menor que o DPL. (o RPL não é utilizado nesse caso). O CPL não muda com a transferência.
        - Para códigos de segmento conforming, o DPL representa o menor número de privilégio (privilégio máximo) que um procedimento de origem pode ter para fazer uma chamada para o segmento de destino

Call Gates, Trap Gates, Interrupt Gates, Task Gates

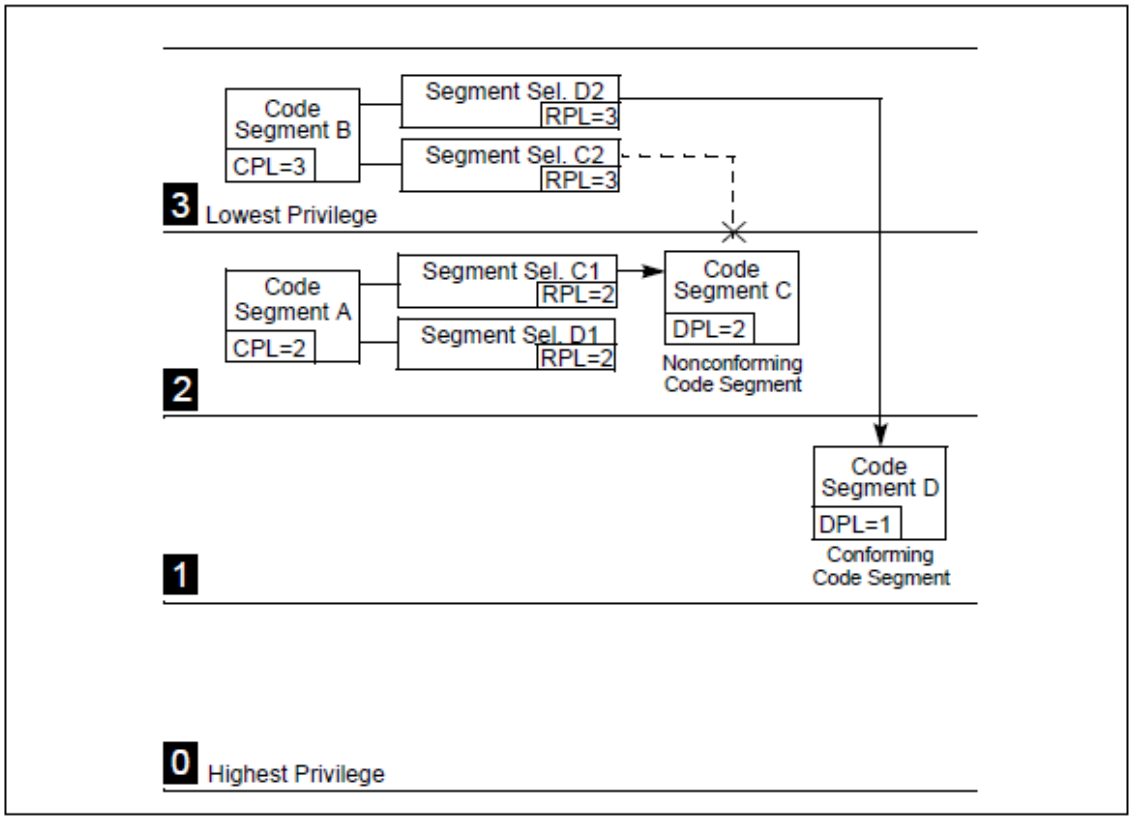

Figure 5-7. Examples of Accessing Conforming and Nonconforming Code Segments From Various Privilege Levels

- A maioria dos segmentos de código são nonconforming. Para esses segmentos, o controle de programa só pode ser transferido entre segmentos com o mesmo nível de privilégio, a menos que a transferência ocorra através de um call gate.

Call Gates, Trap Gates, Interrupt Gates, Task Gates

- Descritores do tipo Gate:
  - Call gates
  - Trap Gates
  - Interrupt Gates
  - Task Gates
  - Task gates são usados para multitask (serão vistos posteriormente)
  - Trap e Interrupt Gates são tipos especiais de call gates usados para tratamento de exceções e interrupções (serão vistos posteriormente)
  - Iremos tratar agora de Call Gates.

Call Gates

- Controlam a transferência de execução entre níveis de privilégio diferentes
- Também pode ser usado para transferir controle entre segmentos de 16-bit e 32-bit.
- Um descritor do tipo Call Gate pode existir na GDT ou em uma LDT.
    - São seis as suas funções:
        1) Especifica o segmento de código a ser acessado (seletor de segmento).
        2) Especifica o ponto de entrada para um procedimento no segmento de código especificado (offset “do main”)
        3) Especifica o nível de privilégio requerido do caller que tenta acessar o procedimento
        4) Se ocorrer uma troca de pilha, especifica o número de parâmetros que deve ser copiado entre as pilhas
        5) Define o tamanho dos valores a serem adicionados na pilha de destino
            - Gates de 16(32) bits forçam pushes de 16(32)-bits.
        6) Especifica se o descritor é válido.

Call Gates

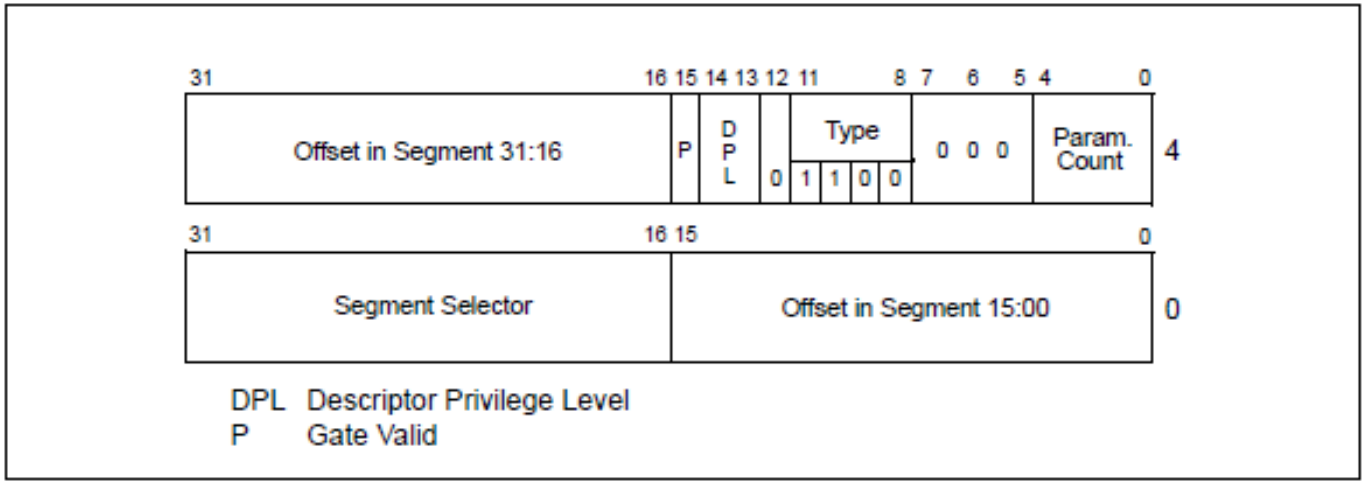

Figure 5-8. Call-Gate Descriptor

- Controlam a transferência de execução entre níveis de privilégio diferentes
- Também pode ser usado para transferir controle entre segmentos de 16-bit e 32-bit.

Acessando um Call Gate

- Para acessar um call gate, um far pointer para o gate é fornecido como o operador de destino em uma instrução CALL ou JMP. O seletor de segmento desse pointer identifica o call gate; o offset é necessário, mas não é usado pelo processador (pode ser qualquer valor)

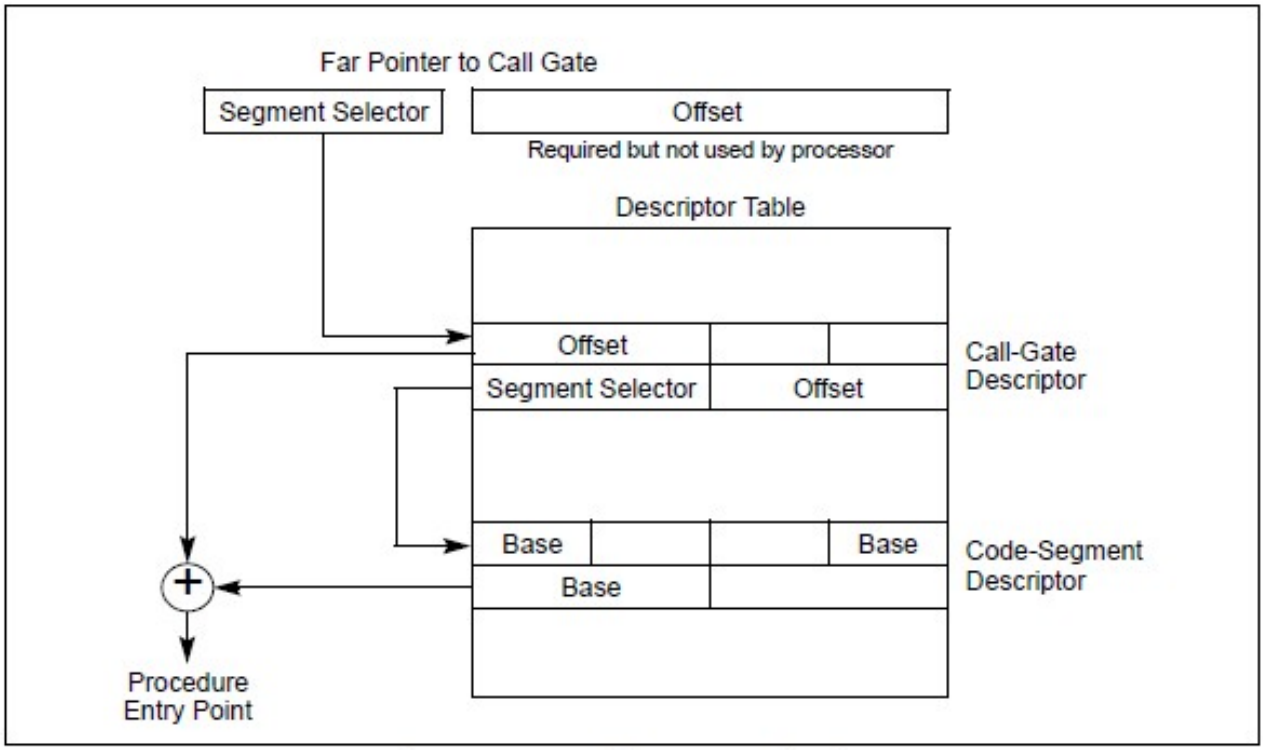

Figure 5-10. Call-Gate Mechanism

Controle de acesso em um Call Gate

- Quatro informações são utilizadas para checar a validade da transferência de controle de um programa através de um call gate:
  - O CPL (nível atual de privilégio)
  - O RPL do seletor de segmento do call gate
  - O DPL do descritor do call gate
  - O DPL do descritor de segmento para o qual o call gate aponta

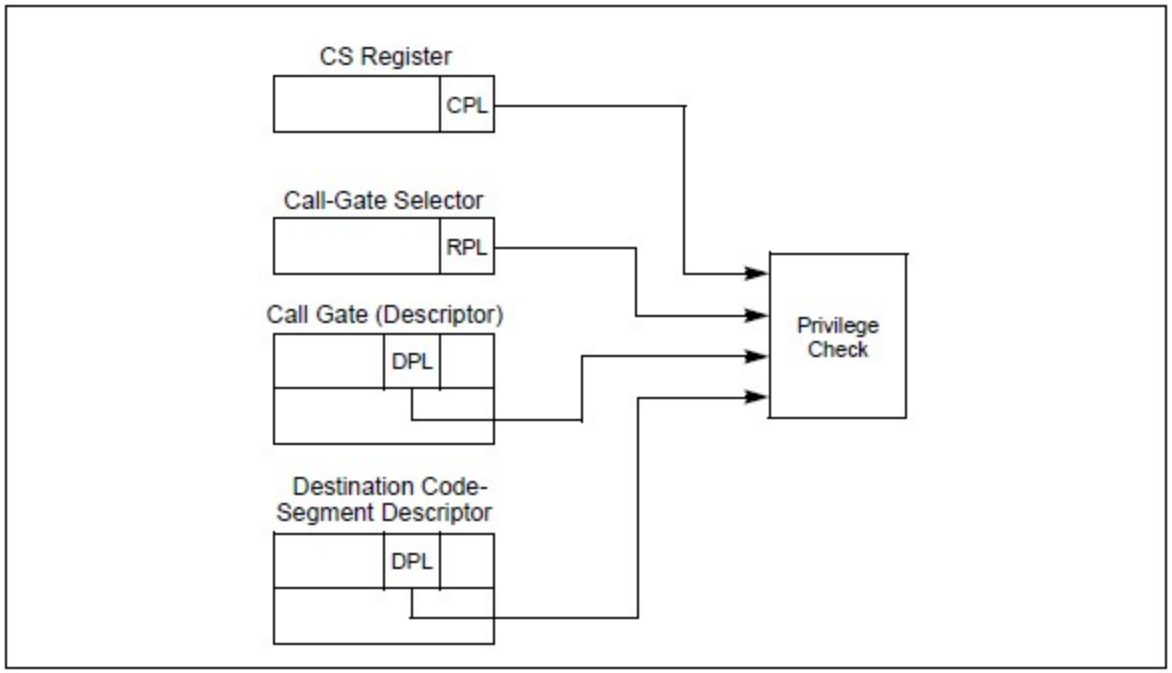

Figure 5-11. Privilege Check for Control Transfer with Call Gate

Regras de acesso

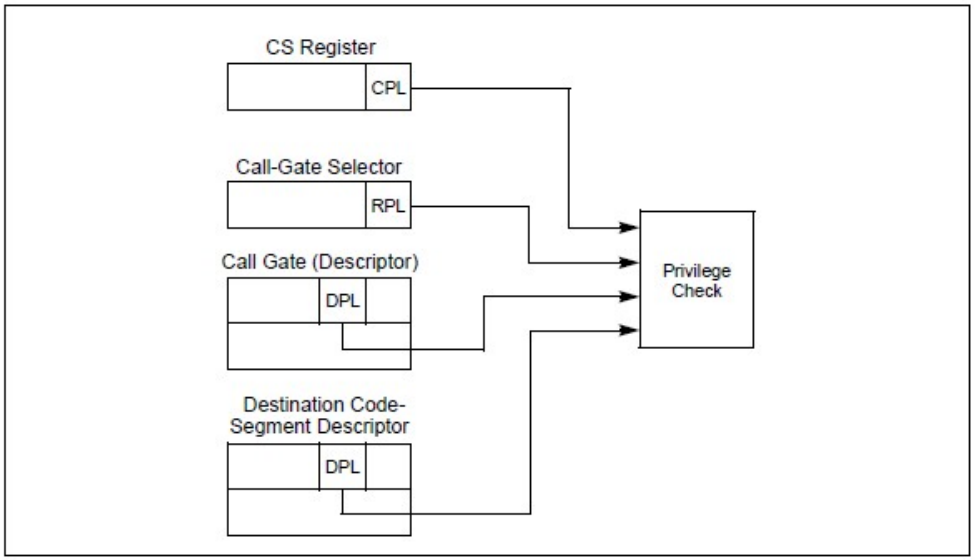

Figure 5-11. Privilege Check for Control Transfer with Call Gate

Table 5-1. Privilege Check Rules for Call Gates

Instruction	Privilege Check Rules
CALL	CPL ≤ call gate DPL; RPL ≤ call gate DPL Destination conforming code segment DPL ≤ CPL Destination nonconforming code segment DPL ≤ CPL
JMP	CPL ≤ call gate DPL; RPL ≤ call gate DPL Destination conforming code segment DPL ≤ CPL Destination nonconforming code segment DPL = CPL

Note que somente a instrução CALL pode mudar para um segmento nonconforming de maior privilégio

Exemplo

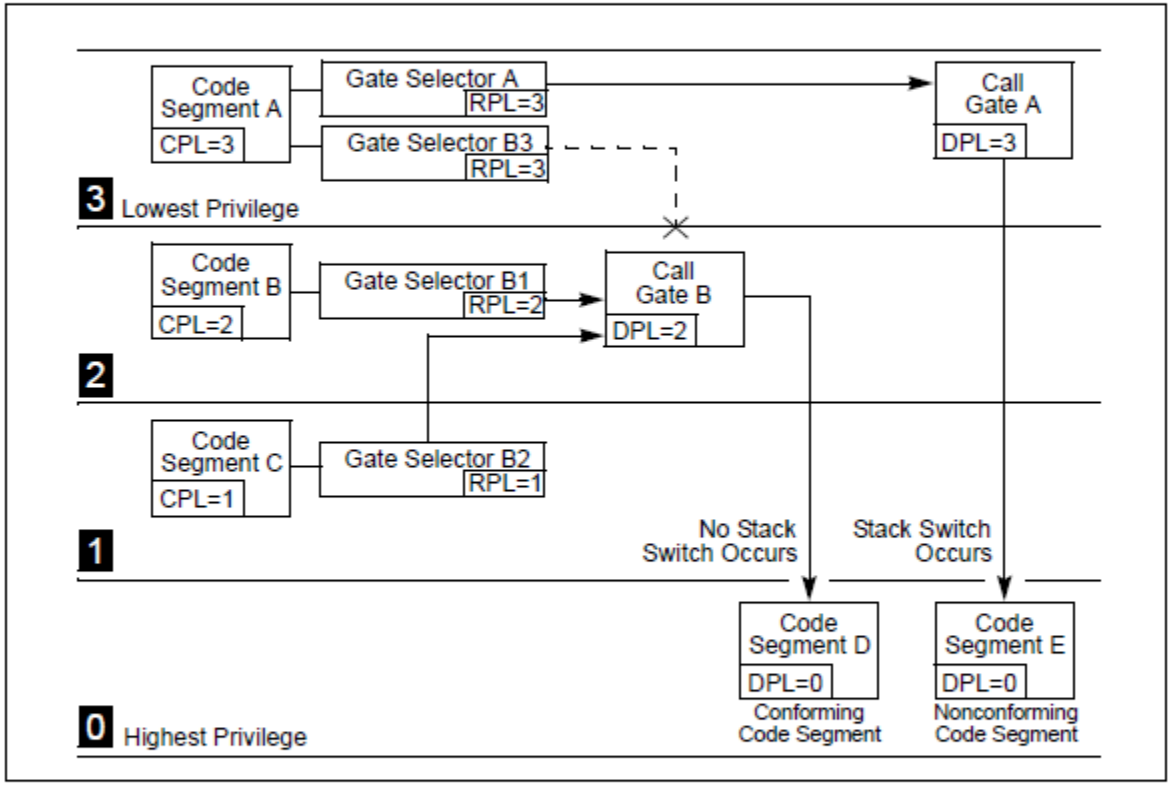

Figure 5-12. Example of Accessing Call Gates At Various Privilege Levels

Se um CALL é feito para um segmento nonconforming de maior privilégio (nível numericamente menor), o CPL é diminuído para o do segmento de código do destino, e OCORRE UMA TROCA DE PILHA.

Exemplos de uso

- Call gates permitem que o mesmo segmento de código tenham procedimentos que o acessem com diferentes níveis de privilégio.
    - Exemplo: o sistema operacional localizado em um segmento de código pode ter serviços que podem ser utilizados tanto por programas de aplicação quando pelo sistema operacional (por exemplo, procedimentos de IO). Podem ser definidos Call gates para esses procedimentos que permitam o acesso em todos os níveis de prioridade (0 até 3). Call gates mais privilegiados (com DPLs 0 ou 1) podem então ser definidos para outros serviços que só devem ser usados pelo sistema operacional (como procedimentos que inicializam drivers de dispositivo).

Troca de Pilha

- Sempre que um call gate é usado para transferir o controle do programa para um segmento nonconforming de maior privilégio (isto é, o DPL do segmento nonconforming de destino é menor que o CPL), o processador automaticamente muda para a pilha do nível de privilégio do código de destino.
- Essa mudança é feita por dois motivos:
    - Evitar que procedimentos de maior privilégio tenham crashed por conta de espaço insuficiente na pilha
    - Evitar que procedimentos de menor privilégio interfiram (por acidente ou propositadamente) em procedimentos de maior privilégio através de uma pilha compartilhada
  - O sistema operacional é responsável por criar pilhas e descritores de pilha para todos os níveis de privilégio que serão usados (por uma tarefa) e por armazenar os apontadores para ela na TSS.
  - Mesmo se o sistema operacional não for multi-tarefa, se ele roda no modo protegido ele é obrigado a criar pelo menos uma TSS para essa função.

Troca de Pilha

- Quando um CALL através de um call gate resulta em uma mudança de nível de privilégio, o processador realiza os seguintes passos para trocar pilhas e começar a execução do procedimento chamado no novo nível de privilégio:

Troca de Pilha

1) Usa o DPL do segmento de código de destino (novo CPL) para selecionar um apontador para a nova pilha (seletor de segmento e stack pointer) a partir da TSS.
2) Lê o seletor de segmento e o stack pointer da pilha de destino a partir da TSS. Qualquer violação de limite detectada ao ler o seletor do segmento de pilha, stack pointer, ou descritor de segmento de pilha causas uma exceção “invalid TSS”.
3) Checa o descritor do segmento de pilha, verificando os privilégios e o tipo. Gera uma “invalid TSS” se forem detectadas violações.
4) Salva temporariamente os valores atuais do SS e do ESP
5) Carrega o seletor de segmento e o stack pointer para a nova pilha nos registradores SS e ESP
6) Empilha na nova pilha dos antigos valores de SS e ESP (salvos em 4)
7) Copia o número de parâmetros especificados da pilha antiga. Se for 0, não copia nenhum parâmetro.
8) Empilha o apontador para a instrução de retorno (CS e EIP) na nova pilha
9) Carrega o seletor de segmento do novo segmento de código e novo instruction pointer no CS e EIP, e começa a execução do código chamado.

Troca de Pilha

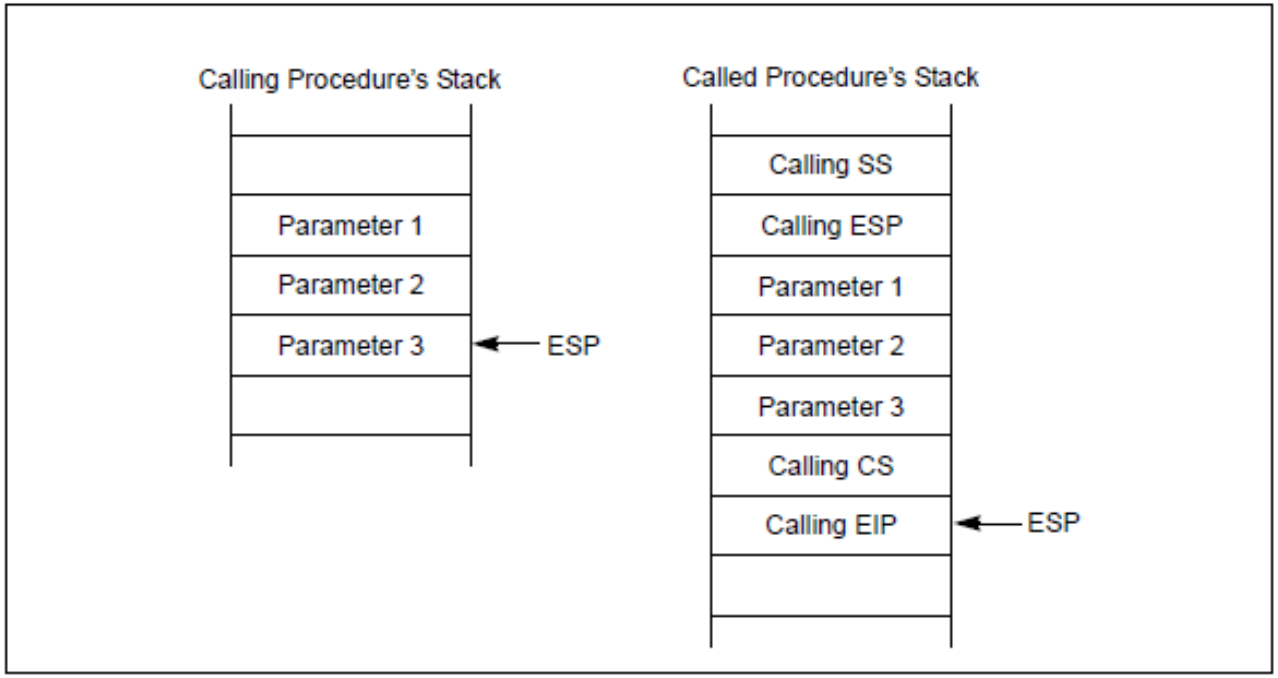

Figure 5-13. Stack Switching During an Interprivilege-Level Call

Interrupt Gates e Trap Gates

Interrupt Gates e Trap Gates são partes do mecanismo de tratamento de interrupções e exceções do modo protegido

- Isto é: ocorreu uma condição que exige a atenção do processador
- **Interrupções**: em resposta a sinais do hardware (ex.: periféricos); requisições de software (instruções INT n).
- **Exceções**: o processador detecta uma condição de erro enquanto executa uma instrução. Ex.: divisão por zero, violação de proteção, page faults, erros internos do hardware.

Exceções

Exceções podem ser de 3 tipos: **faults**, **traps** e **aborts**.

- **Fault:** uma exceção que geralmente pode ser corrigida e que, uma vez corrigida, permite que o programa seja reiniciado sem perda de continuidade. Quando ocorre uma fault, o processador restaura o estado da máquina para aquele anterior ao início da instrução que gerou uma fault. O endereço de retorno para o fault handler (conteúdo de CS e EIP) aponta para a instrução que gerou a fault, ao invés da instrução que a segue.
  - Exemplo: Page fault

Exceções

Exceções podem ser de 3 tipos: **faults**, **traps** e **aborts**.

- **Trap:** gerada imediatamente em seguida a instrução que a causa. Permite que o programa seja continuado sem perda de continuidade. Quando ocorre uma trap, o endereço de retorno do trap handler aponta para a instrução seguinte àquela que gerou a trap.
  - Exemplo: Overflow (instrução INTO)

Exceções

Exceções podem ser de 3 tipos: **faults**, **traps** e **aborts**.

- **Abort:** exceção que não aponta o local preciso da instrução que causou a exceção e não permite o reinício do programa/task que gerou a exceção.
  - Usado para reportar errors graves, tais como erros de hardware

Interrupt Descriptor Table (IDT)

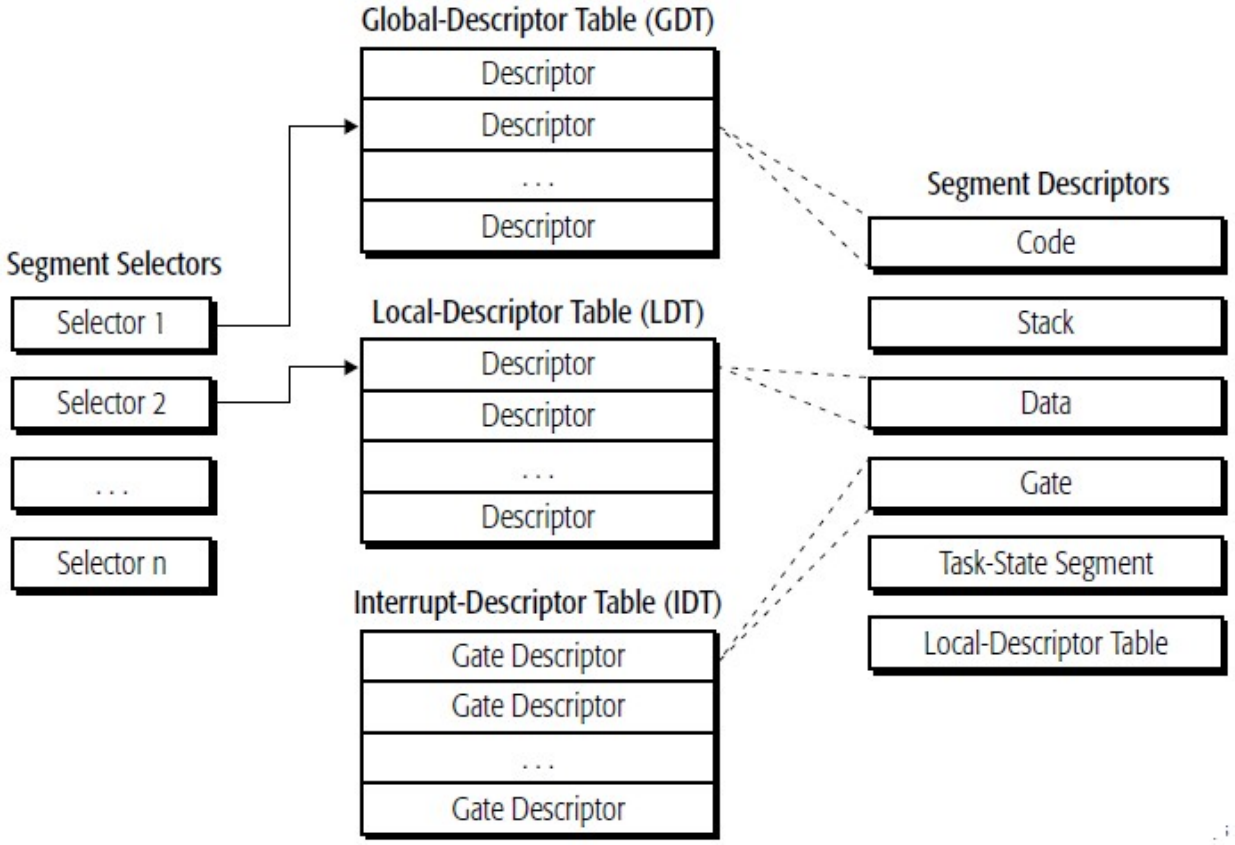

Figure 4-1. Segmentation Data Structures

Interrupt Descriptor Table (IDT)

Contém descritores do tipo “gate” para tratadores de exceções/interrupções.

- O tratamento pode ser feito na mesma task
  - Interrupt gates e Trap gates
  - Ou em outra task
  - Task gates

Índices da IDT

Table 6-1. Protected-Mode Exceptions and Interrupts

Vector No.	Mnemonic	Description	Type	Error Code	Source
0	#DE	Divide Error	Fault	No	DIV and IDIV instructions.
1	#DB	RESERVED	Fault/ Trap	No	For Intel use only.
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.[1]
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.[2]
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.

Índices da IDT (cont.)

Table 6-1. Protected-Mode Exceptions and Interrupts (Contd.)

17	#AC	Alignment Check	Fault	Yes (Zero)	Any data reference in memory.[3]
18	#MC	Machine Check	Abort	No	Error codes (if any) and source are model dependent.[4]
19	#XM	SIMD Floating-Point Exception	Fault	No	SSE/SSE2/SSE3 floating-point instructions[5]
20-31	—	Intel reserved. Do not use.			
32-255	—	User Defined (Non-reserved) Interrupts	Interrupt		External interrupt or INT *n* instruction.

NOTES:

1. The UD2 instruction was introduced in the Pentium Pro processor.
2. Processors after the Intel386 processor do not generate this exception.
3. This exception was introduced in the Intel486 processor.
4. This exception was introduced in the Pentium processor and enhanced in the P6 family processors.
5. This exception was introduced in the Pentium III processor.

O IDTR

O Interrupt Descritor Table Register (IDTR) guarda o endereço de base (32-bits) e o limite (16 bits) da IDT.

- A instrução LIDT (load IDT register) carrega o IDTR com um endereço base e limite presentes em um endereço de memória. Essa instrução só pode ser executada quando o CPL é 0. É normalmente usada pela inicialização do SO. O SO pode também utilizá-la para trocar a IDT.
- A instrução SIDT copia o valor da base e limite guardados no IDTR para a memória. Pode ser executada em qualquer nível de privilégio.

O IDTR

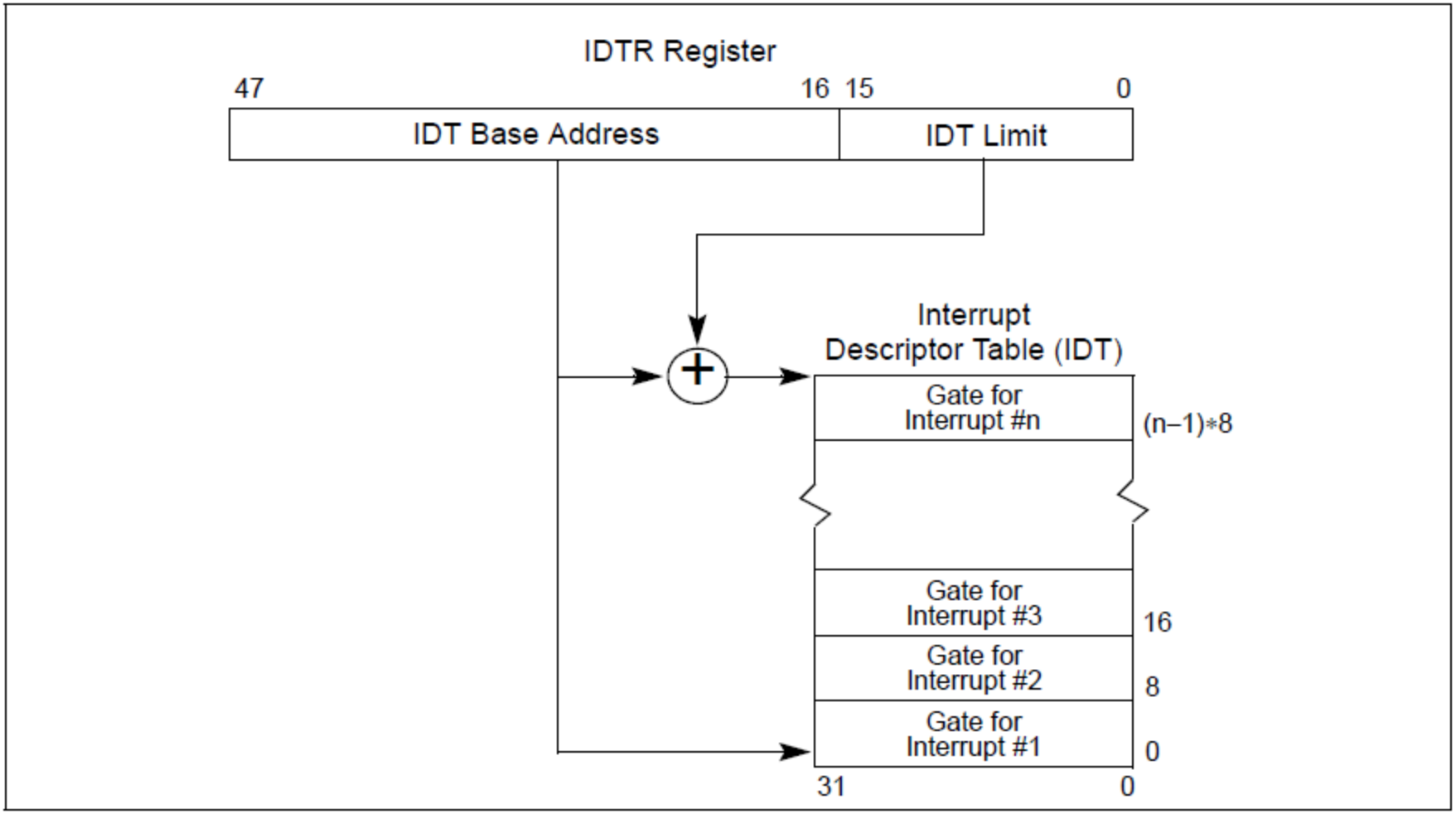

Figure 6-1. Relationship of the IDTR and IDT

Interrupt Gates e Trap Gates

- Se o exception/interrupt handler para o índice gerado for um interrupt gate descriptor ou um trap gate descriptor, o processador trata a exceção/interrupção de forma semelhante a um CALL para um call gate.
- Se o exception/interrupt handler para o índice gerado for um task gate descriptor, o processador faz um task switch para a tarefa do handler de forma similar a um CALL para um task gate (veremos posteriormente)

Interrupt Gate and Trap Gate Descriptors

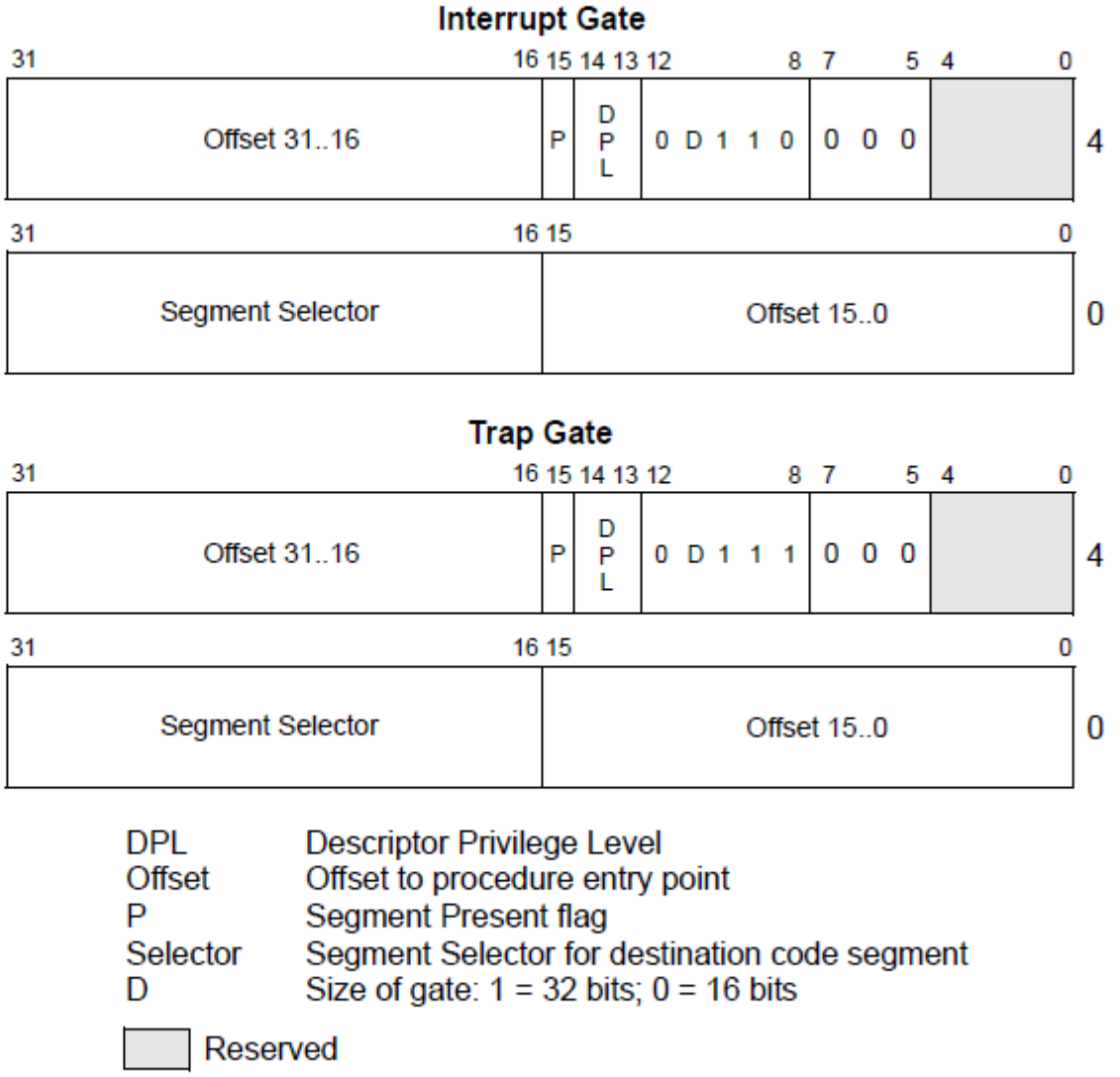

Figure 6-2. IDT Gate Descriptors

Interrupt Gate and Trap Gate Descriptors

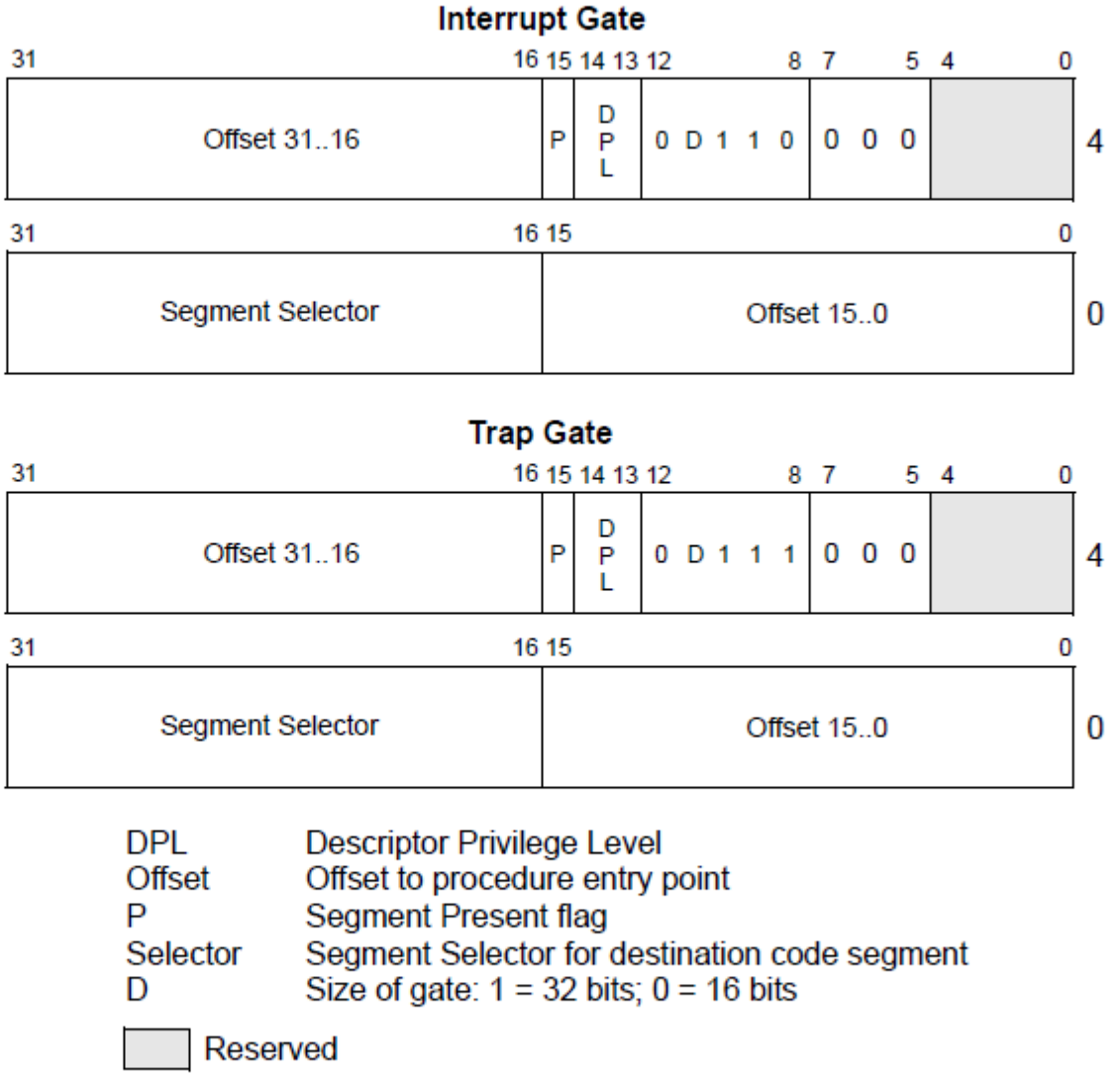

Figure 6-2. IDT Gate Descriptors

Tratamento de exceção/Interrupção

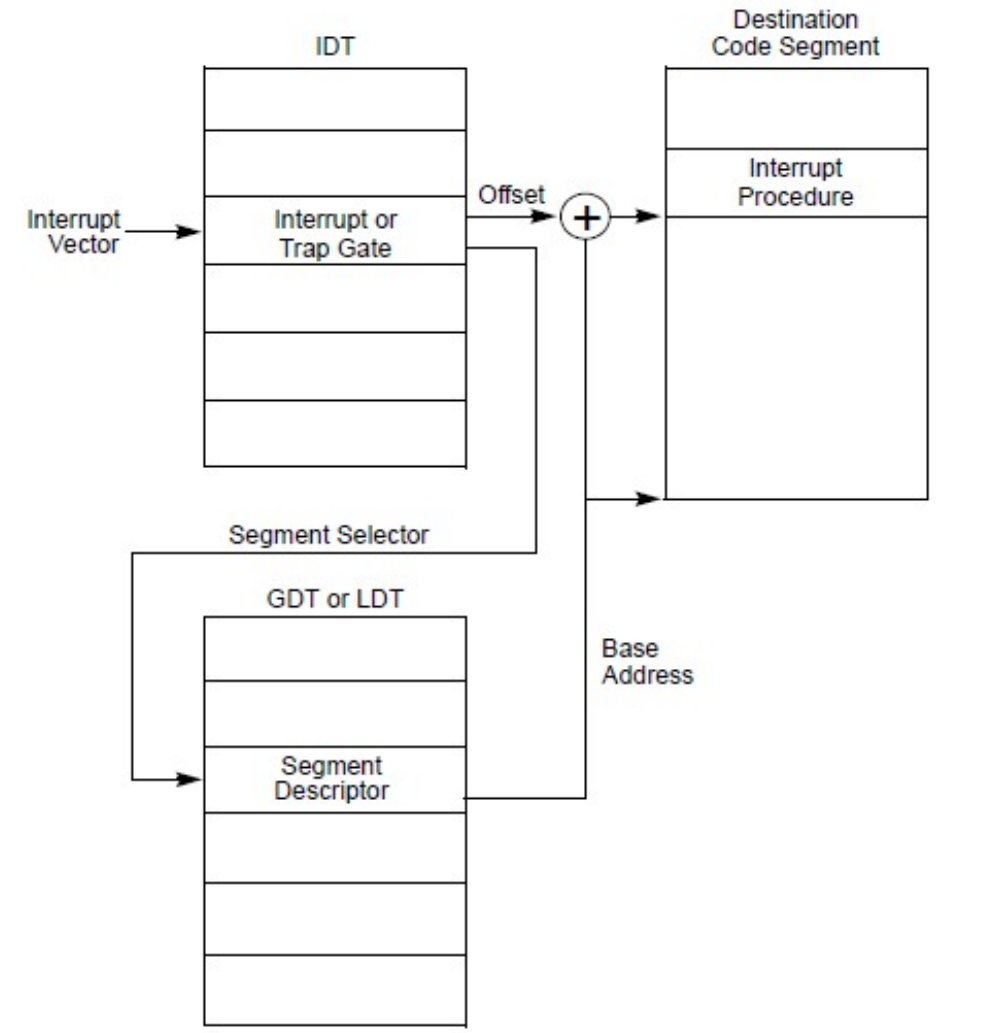

Figure 6-3. Interrupt Procedure Call

Troca de pilha

- Quando o procedimento de tratamento de interrupção/exceção vai ocorrer em um nível de privilégio mais alto (numericamente menor), ocorre uma troca de pilha.
  - Não é possível a transferência de execução para um exception/interrupt handler com nível de privilégio menor (numericamente mais alto) do que o CPL

Troca de pilha

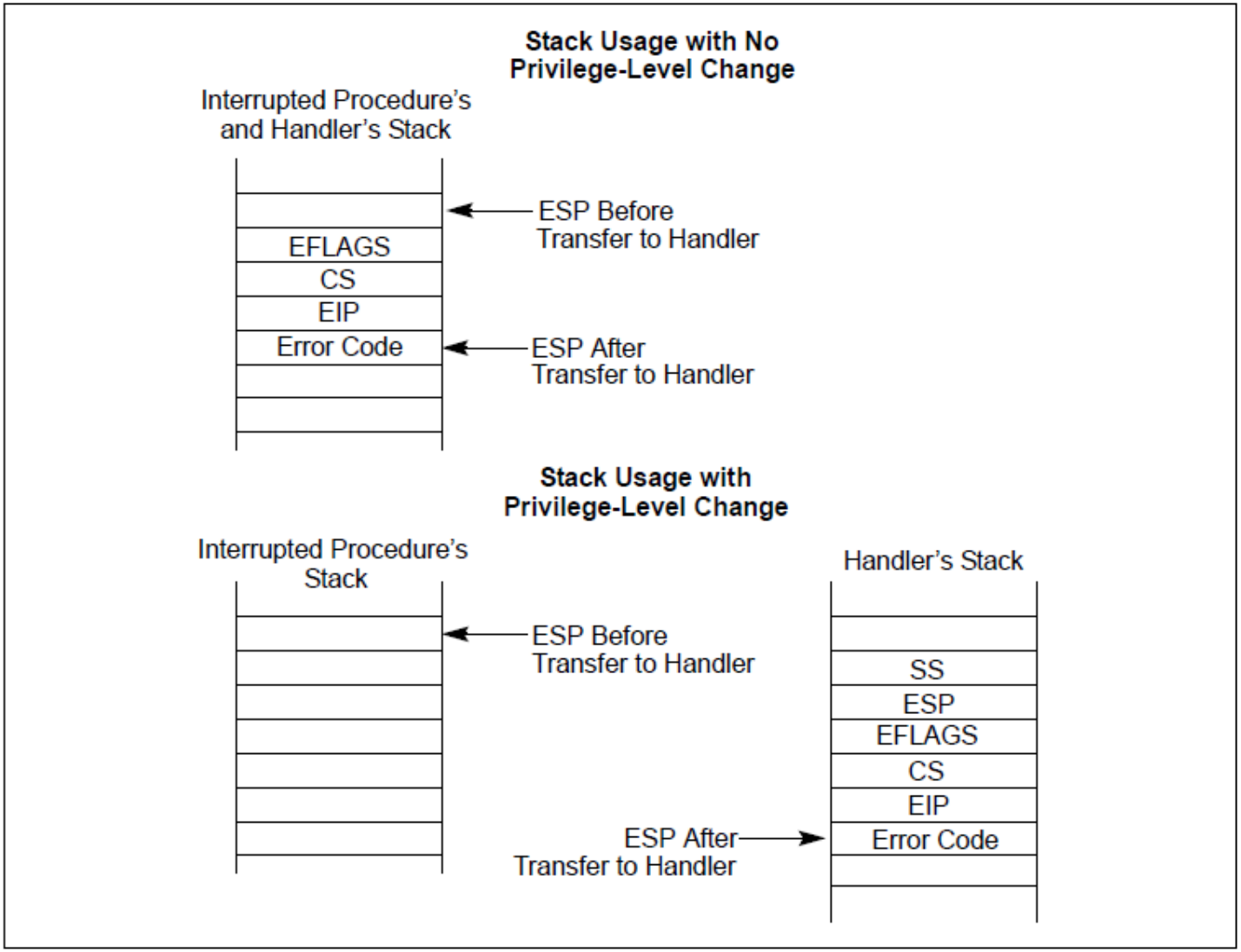

Figure 6-4. Stack Usage on Transfers to Interrupt and Exception-Handling Routines

Interrupt gates x Trap gates

- A diferença entre um interrupt gate e um trap gate é na forma que o processador trata a flag IF no registrador EFLAGS.
- Interrupt gate: o processador faz IF = 0 (interrupt flag) para impedir que outras interrupções (mascaráveis) interfiram com o handler atual. Quando é feito um IRET, o processador restaura o valor do flag.
- Trap gate: não afeta o IF

Multitarefa (Task switch)

Suporte de hardware para execução de múltiplas tarefas (em 32 bits)

- O suporte de hardware tem uso opcional, é possível implementar multi-tarefa via software

Task: unidade de trabalho que um processador pode inicializar (dispatch), executar e suspender.

Em modo 64 bits, o suporte de hardware para task switching não está disponível.

Durante troca de contexto, dados da tarefa guardado em Task State Segment (TSS)

Descritores guardados na GDT

TSS em uso guardado no task register (TR)

Task

Composta por 2 partes:

- Espaço de execução da tarefa (task execution space)
  - Segmento de código; Segmento de pilha; Um ou mais segmentos de dados
  - Pilha separada para cada nível de privilégio usado
- Task-state segment (TSS)
  - Armazena informações sobre o estado da tarefa

Task

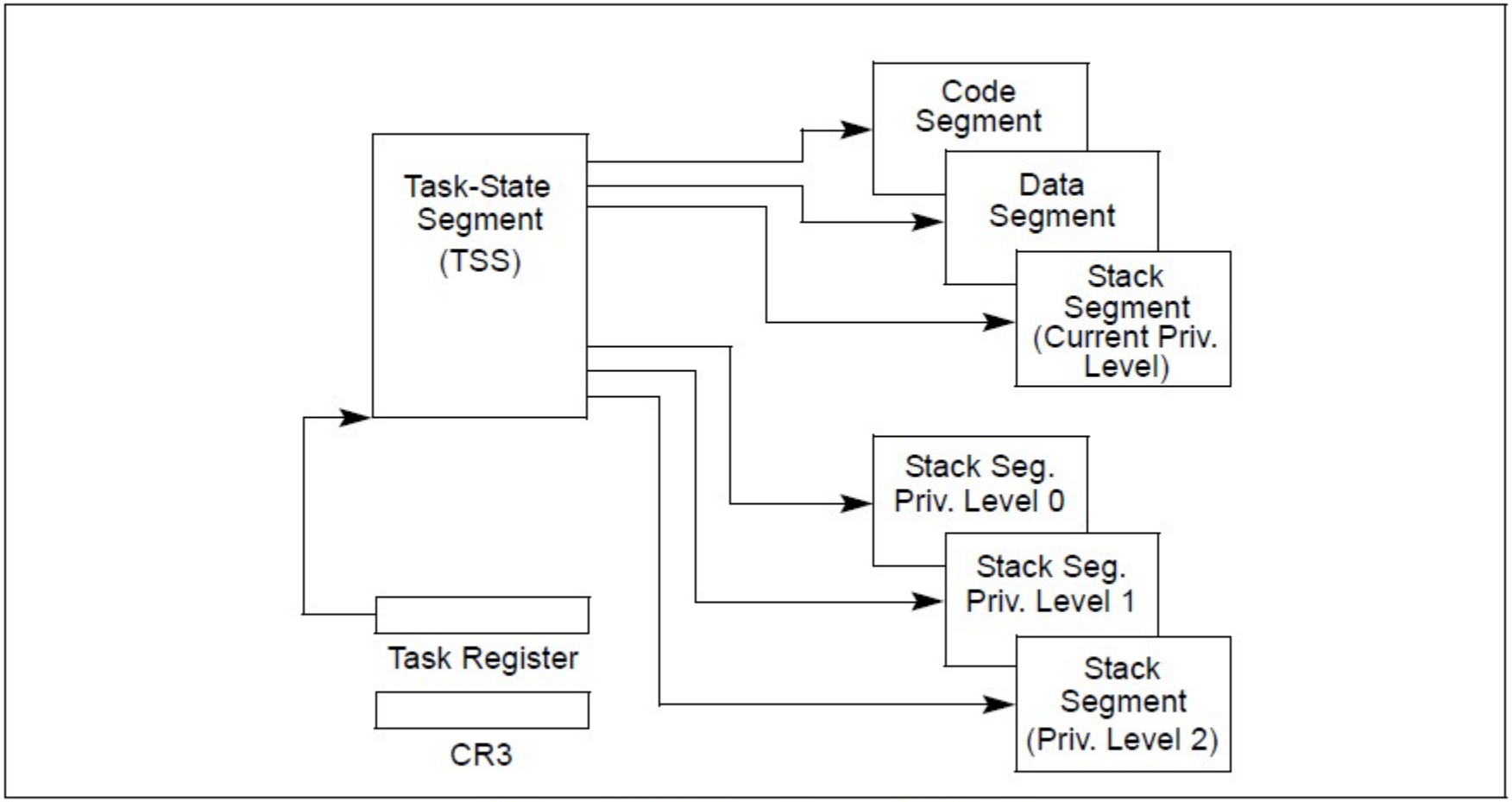

Figure 7-1. Structure of a Task

Task

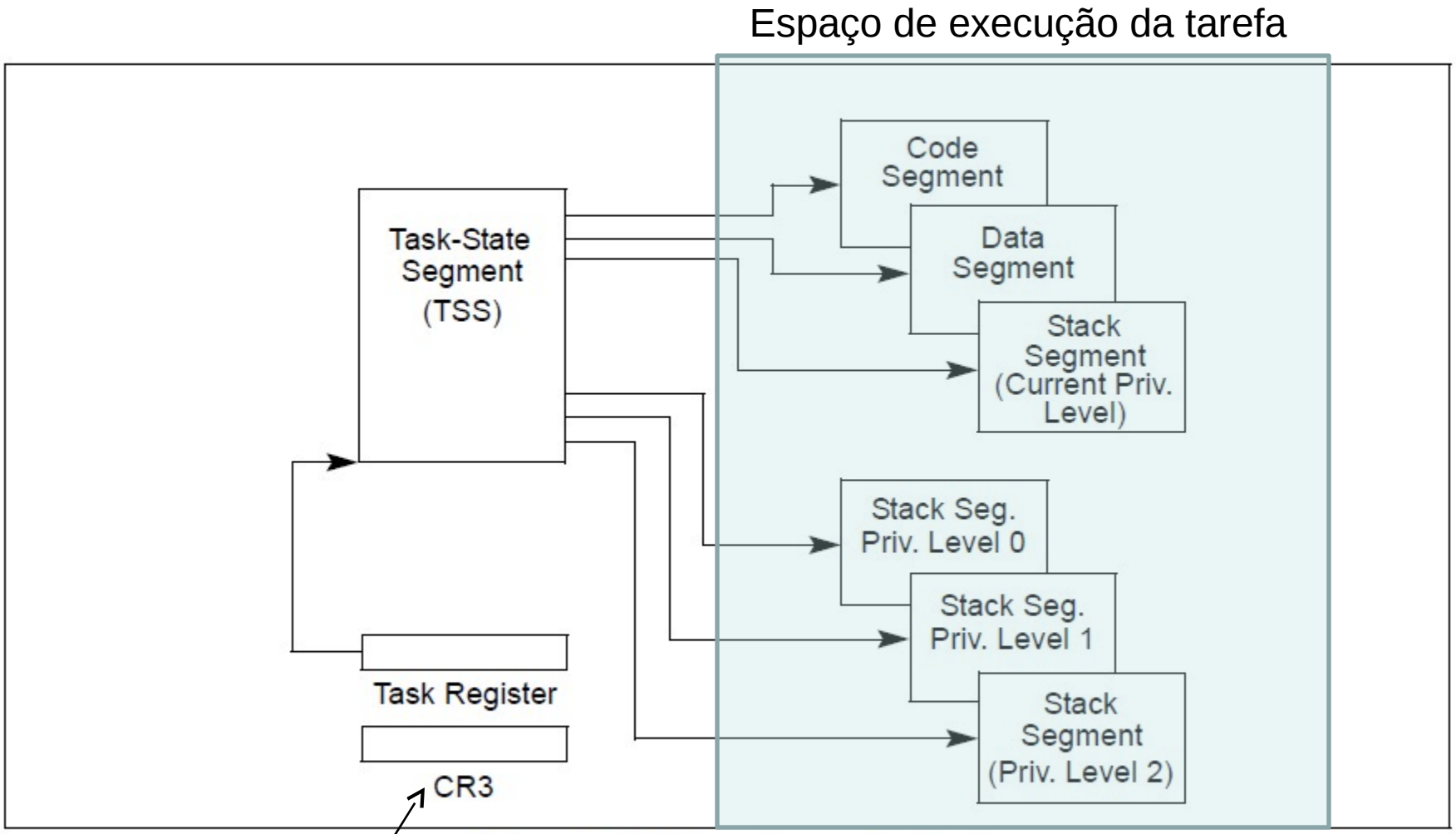

Figure 7-1. Structure of a Task

Usado quando paginação está ativa

Task State Segment

- Guarda todos os dados necessários para restaurar a execução de uma tarefa
- É definido por um descritor de segmento específico, chamado descritor TSS.
  - Que só podem ser colocados na GDT
- Quando operando em modo protegido, é necessário criar ao menos um TSS (para uma tarefa), e o seletor de segmento para a TSS deve ser carregado no Task Register (usando a instrução LTR)

Task State Segment (32 bit)

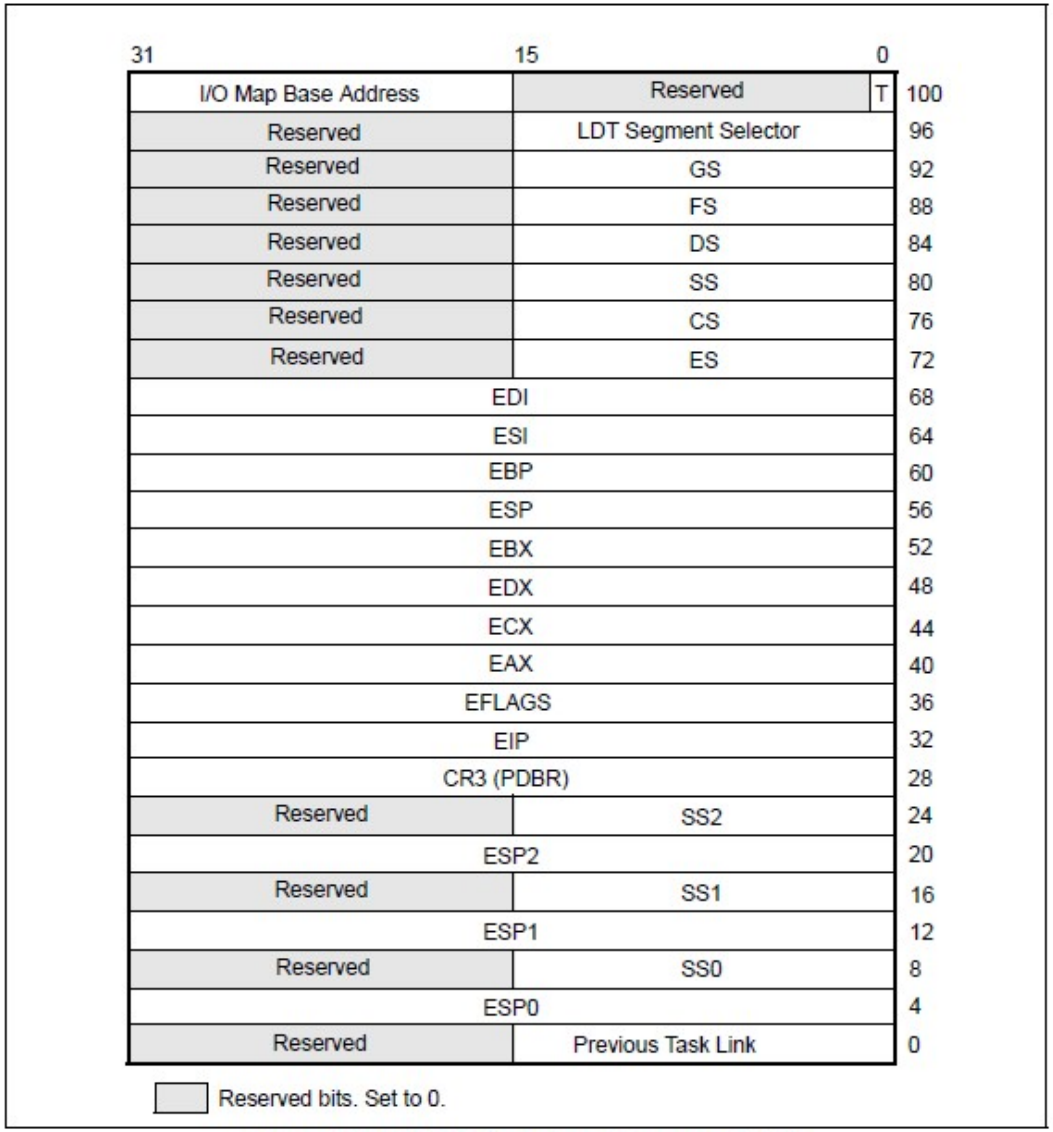

Figure 7-2. 32-Bit Task-State Segment (TSS)

TSS Descriptor

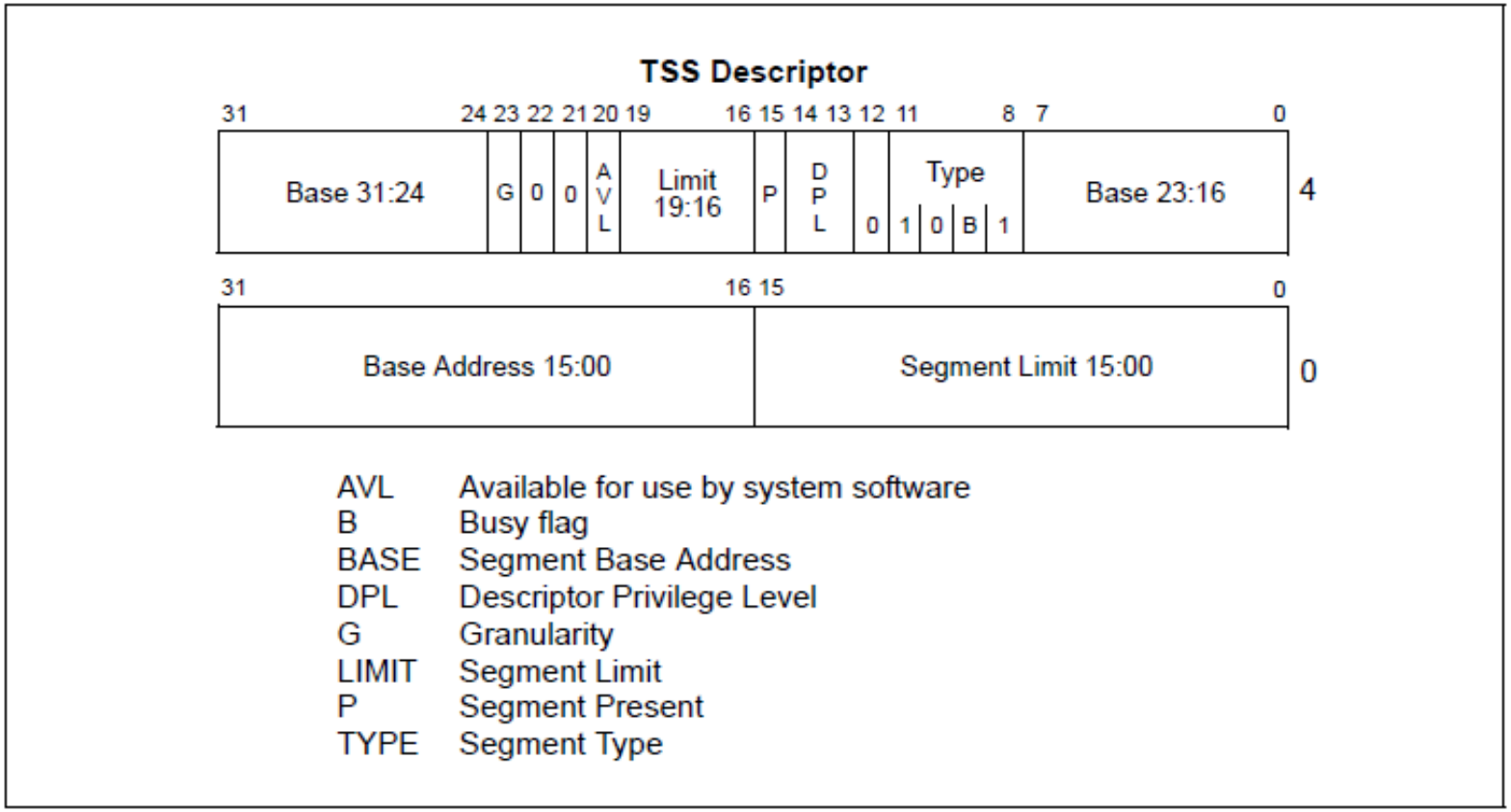

Figure 7-3. TSS Descriptor

Task Register

- Guarda o seletor de segmento para a TSS da task atual.
- As instruções LTR (load task register) e STR (store task register) carregam e leem, respectivamente, a parte visível do task register.
- LTR só pode ser executada quando CPL atual é 0.

Task Register

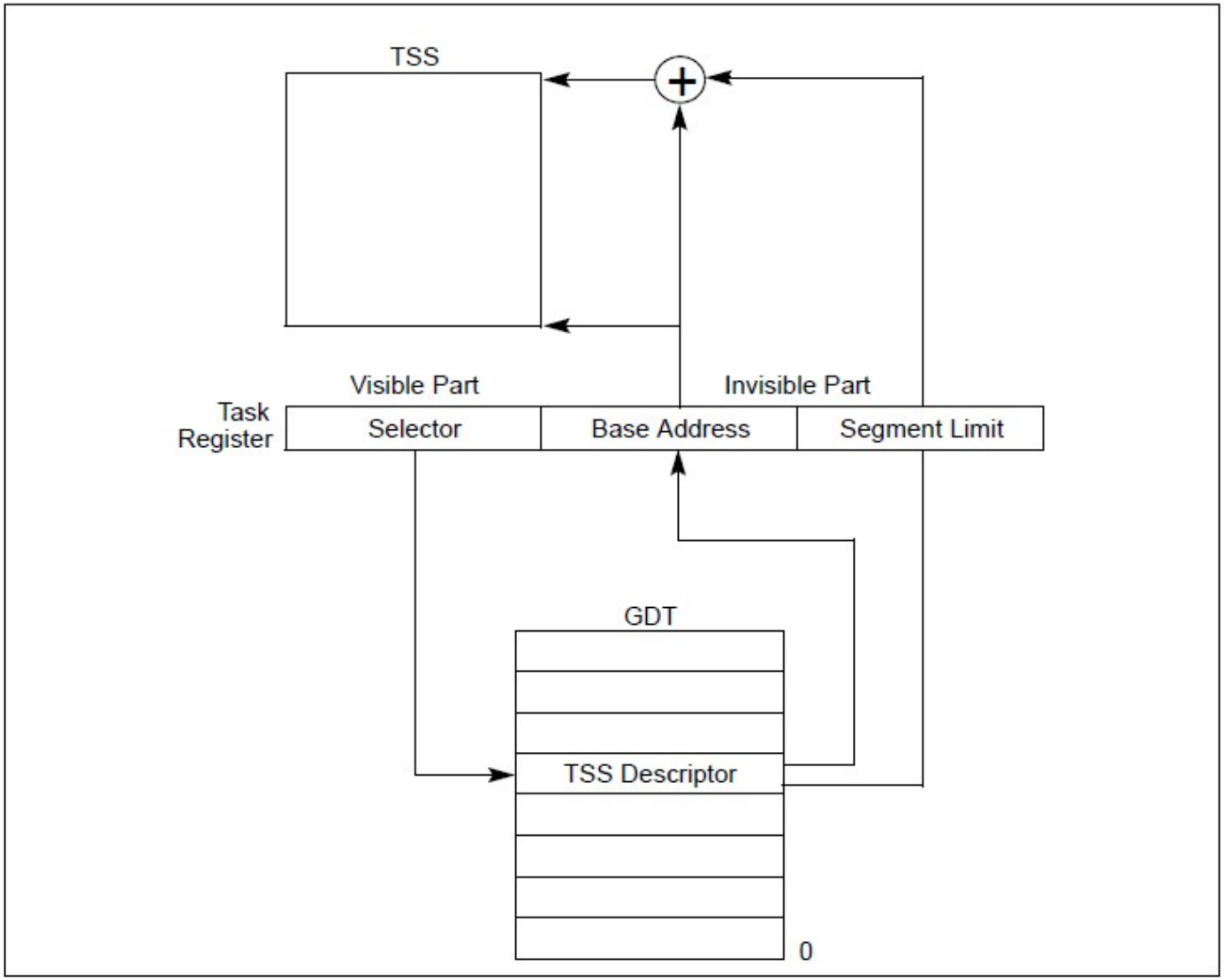

Figure 7-5. Task Register

Task-Gate Descriptor

- Permite o acesso indireto, protegido, a uma task.
- Pode ser colocado na GDT, LDT ou IDT
- Aponta para um TSS descriptor (na GDT)
- Quando é usado, o RPL do descritor para o qual ele aponta não é usado.
- Para conseguir acessar a task apontada pelo task-gate descriptor, o CPL e o RPL do seletor que aponta para o gate tem que ter CPL e RPL inferior ao DPL do task-gate descriptor

Task-Gate Descriptor

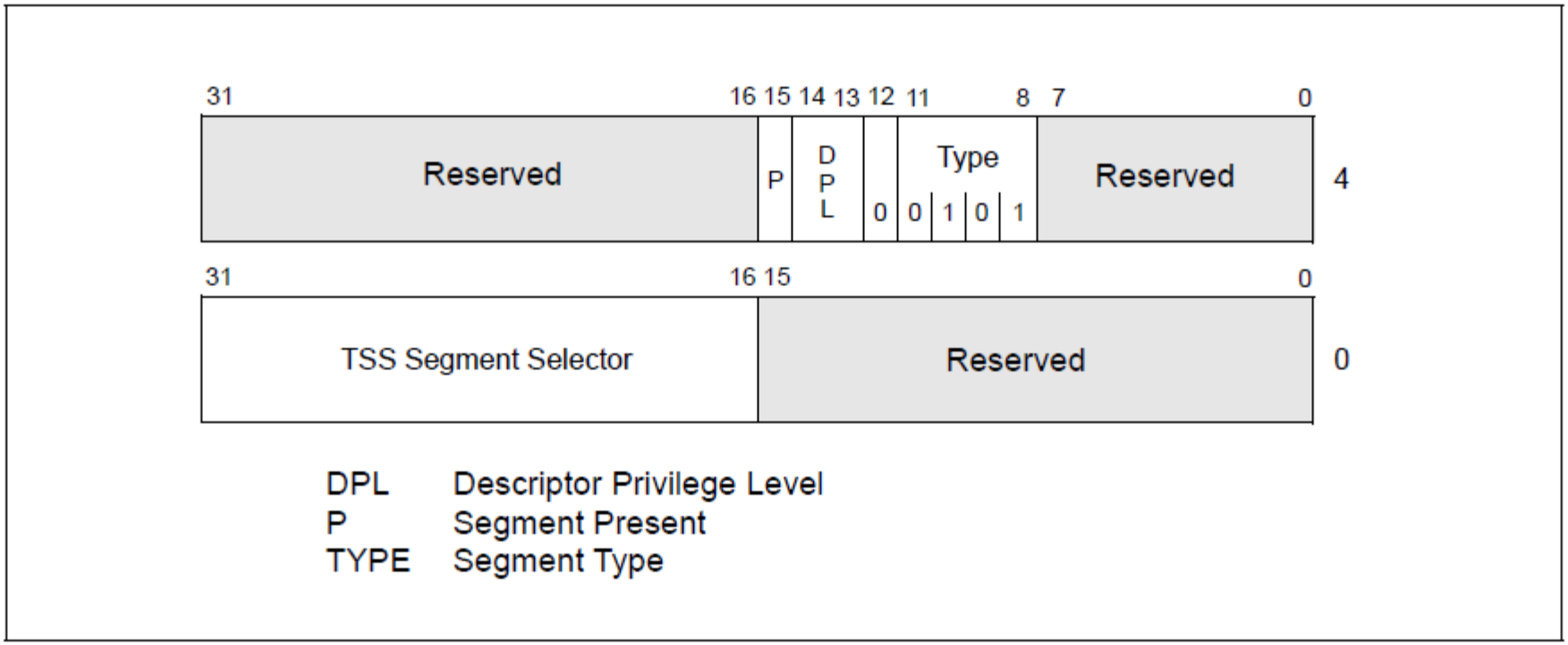

Figure 7-6. Task-Gate Descriptor

Próximos passos...

Criando um bootloader em modo protegido...