X86 - Modo Protegido (32bits)

Introdução
(parte 1)

Modo Real (16 bits) x 32 bits

- Na evolução de 16 bits para 32 bits, necessária para permitir um maior espaço de endereçamento, a Intel não se limitou a expandir os componentes do processador programáveis por software (ex. registradores) para 32 bits.
- Foram introduzidos de forma integral no projeto do processador mecanismos de proteção.

Por que proteção?

- “O objetivo das características de proteção do 80386 é **ajudar a detectar e identificar bugs**. O 80386 suporta aplicações sofisticadas que podem consistir de centenas ou milhares de módulos de programas. Em tais aplicações, a questão é **como bugs podem ser encontrados e eliminados o mais rápido possível e como o seus danos podem ser confinados**. Para **ajudar a depurar aplicações mais rapidamente e torná-las mais robustas em produção**, o 80386 contém mecanismos para **verificar acessos à memória e execução de instruções quanto à conformidade a critérios de proteção**. Esses mecanismos **podem ser usados ou ignorados**, de acordo com objetivos do design do sistema”.

  INTEL 80386 PROGRAMMER'S REFERENCE MANUAL, 1986

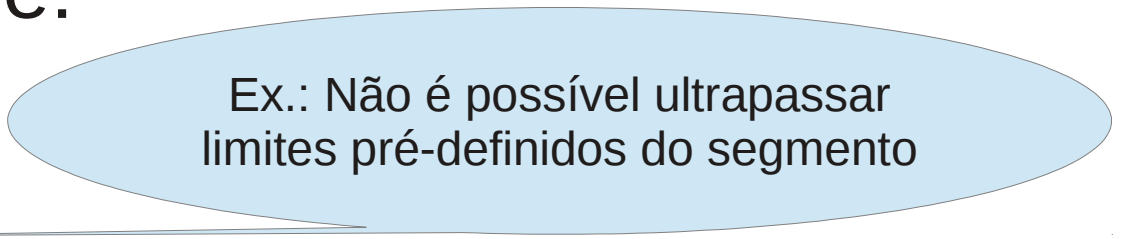
Aspectos de proteção

- Verificação de:
  - Tipo
  - Limites
- Restrições de:
  - Endereços acessíveis
  - Pontos de entrada de procedimentos
  - Conjunto de instruções disponíveis

Ex.: Jumps devem ser feitos para segmentos de código; segmentos de código são apenas leitura

Aspectos de proteção

- Verificação de:
  - Tipo
  - Limites
- Restrições de:
  - Endereços acessíveis
  - Pontos de entrada de procedimentos
  - Conjunto de instruções disponíveis



Aspectos de proteção

- Verificação de:
  - Tipo
  - Limites
- Restrições de:
  - Endereços acessíveis
  - Pontos de entrada de procedimentos
  - Conjunto de instruções disponíveis

Utilização de níveis de privilégio

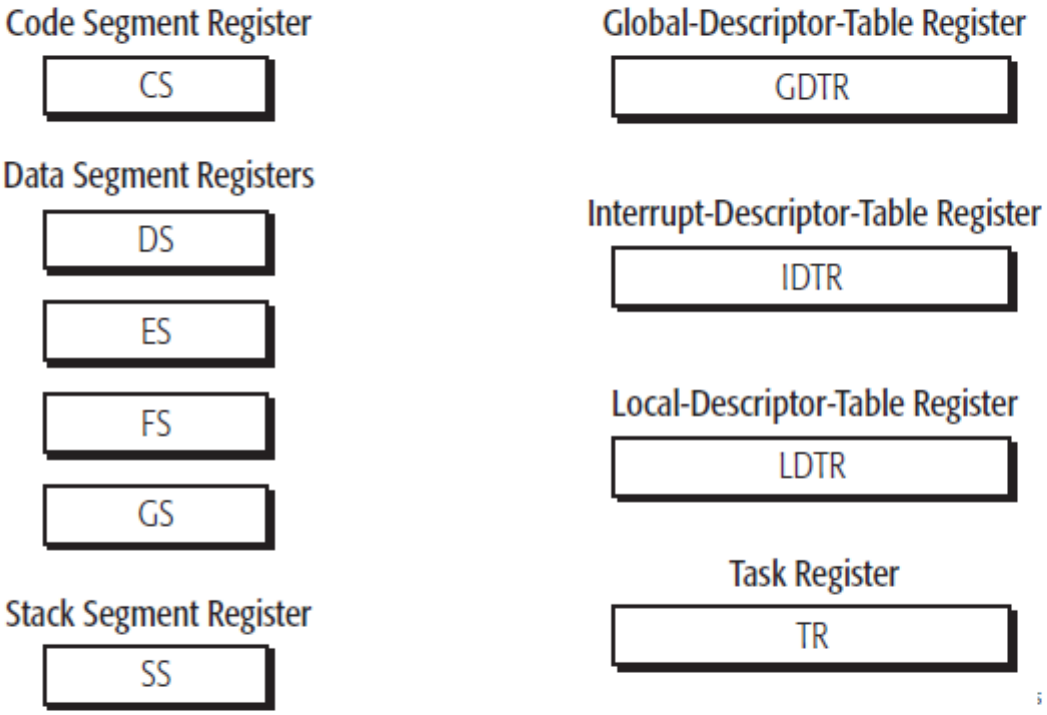
Modo Protegido

- **Características Gerais**
  - 4 GB
  - Multitarefa
  - Memória Virtual
  - Gerenciamento e Proteção de Memória
  - Registradores de 32 bits (EIP,ESP,...)
- **Proteção de Acesso**
- **Tabelas de descritores de segmentos**: IDT,GDT,LDT
- **Registradores de Controle:** CR0,...,CR4
- **Registradores de Depuração:** DR0,...,DR7

Segmentos

- **Segmento: Bloco contínuo de memória de tamanho e localização variáveis**
- **Segmentos do usuário (utilizado pelas aplicações), registradores: CS, DS, SS, ES, FS, GS (code, data, stack, extra, ...)**
- **Novidade do modo protegido:**
  - **Segmentos de sistema: são utilizados para definir atributos das novas funcionalidades trazidas pelo modo protegido. Ex.: controle de acesso, multitarefa.**
  - **Novos registradores, associados aos segmentos de sistema:**
    - **GDTR: Global-descriptor-table register**
    - **LDTR: Local-descriptor-table register**
    - **IDTR: Interrupt-descriptor-table register**
    - **TR: Task register**

Segmentação: registradores



Segmentos

- **Segmentos de sistema contém estruturas de dados inicializadas e utilizadas APENAS pelo sistema operacional**
- **Descritores de segmento contém:**
  - **Endereço Base: aponta para a localização inicial do segmento**
  - **Limite: define o tamanho do segmento**
  - **Atributos: definem as características de proteção do segmento**
- **Com o uso de paginação nos OS modernos, segmentação está em desuso, porém algumas funções da segmentação continuam sendo importantes e utilizadas**

Mecanismo de Segmentação

Registradores de segmento = Seletor de descritor

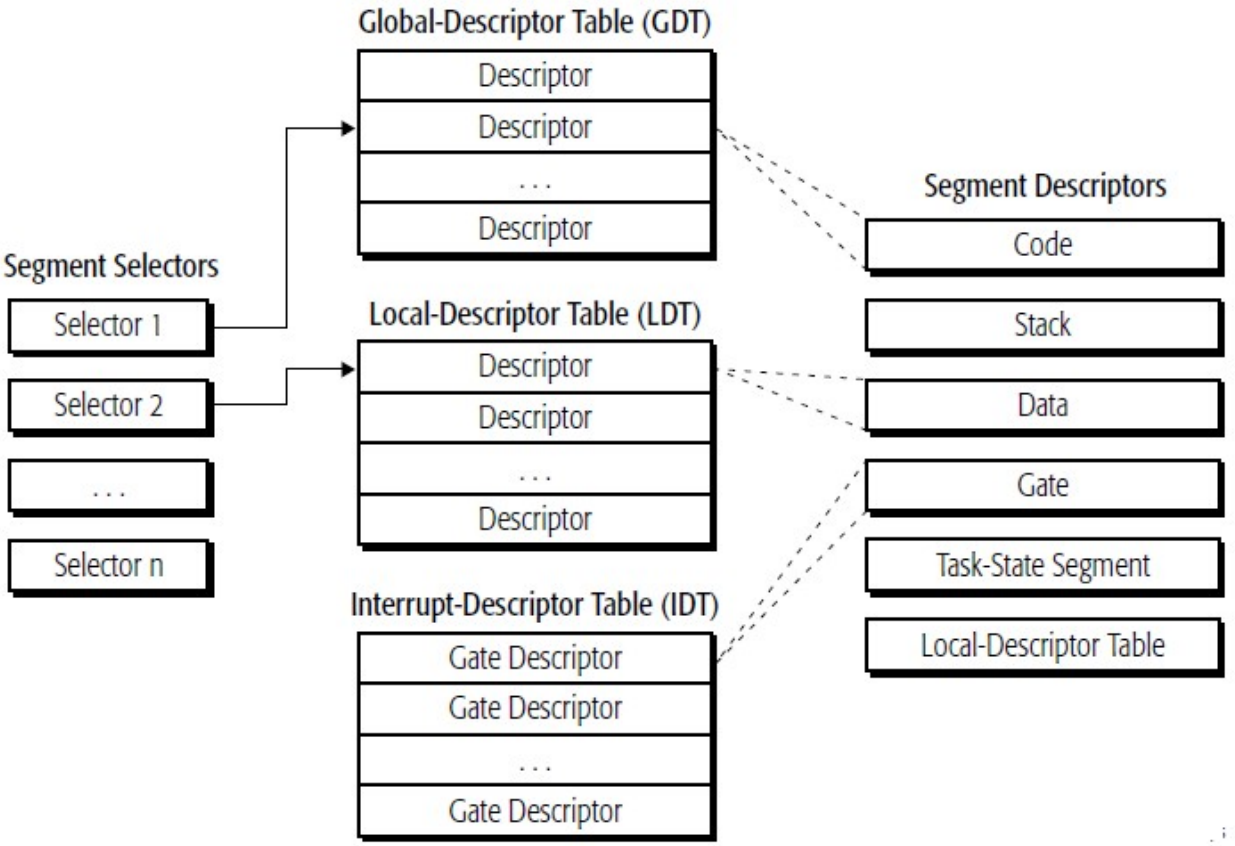


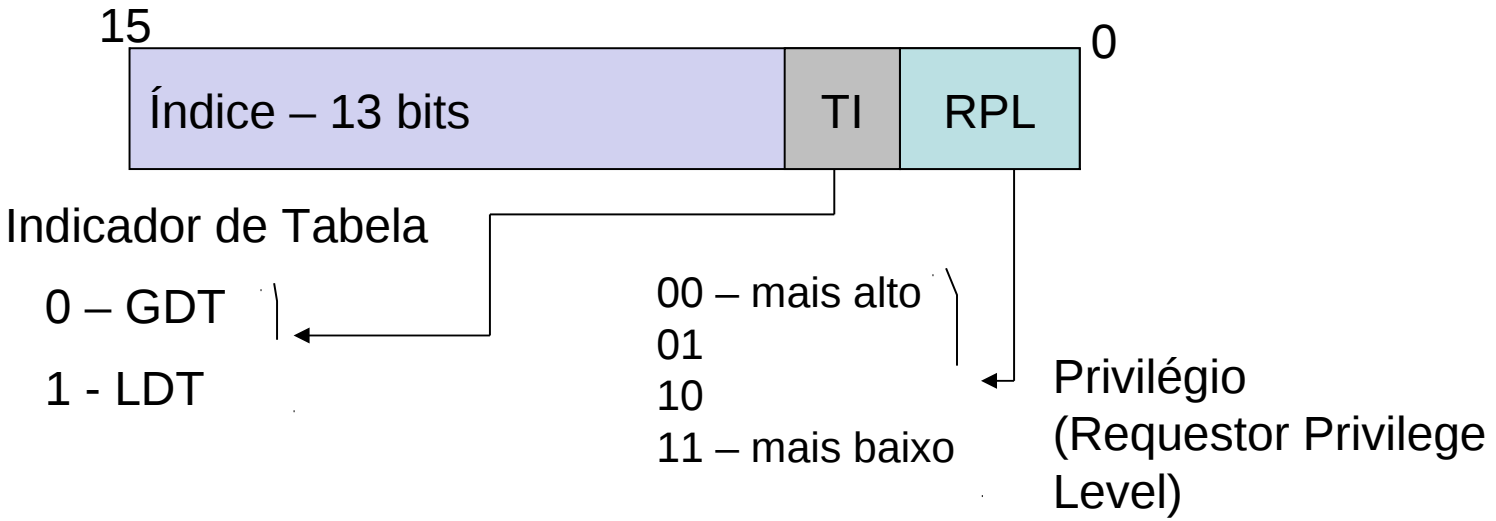
Figure 4-1. Segmentation Data Structures

Segmentação: estrutura de dados e registradores

- **Descritores de segmento: contém a localização do segmento, o seu tamanho, características de proteção, outros atributos**
- **Tabelas de descritores: descritores de segmento são armazenados em uma de três tabelas:**
  - **GDT: Global descriptor table. Segmentos que são compartilhados por todas as tarefas**
  - **LDT: descritores que são usados por tarefas específicas, não são compartilhadas**
  - **TSS: tipo especial de segmento que contém informações quanto ao estado de cada tarefa (exemplo, guarda EFLAGS)**
- **Seletores de segmento: índice para uma tabela de descritores**

Seletor de Segmento

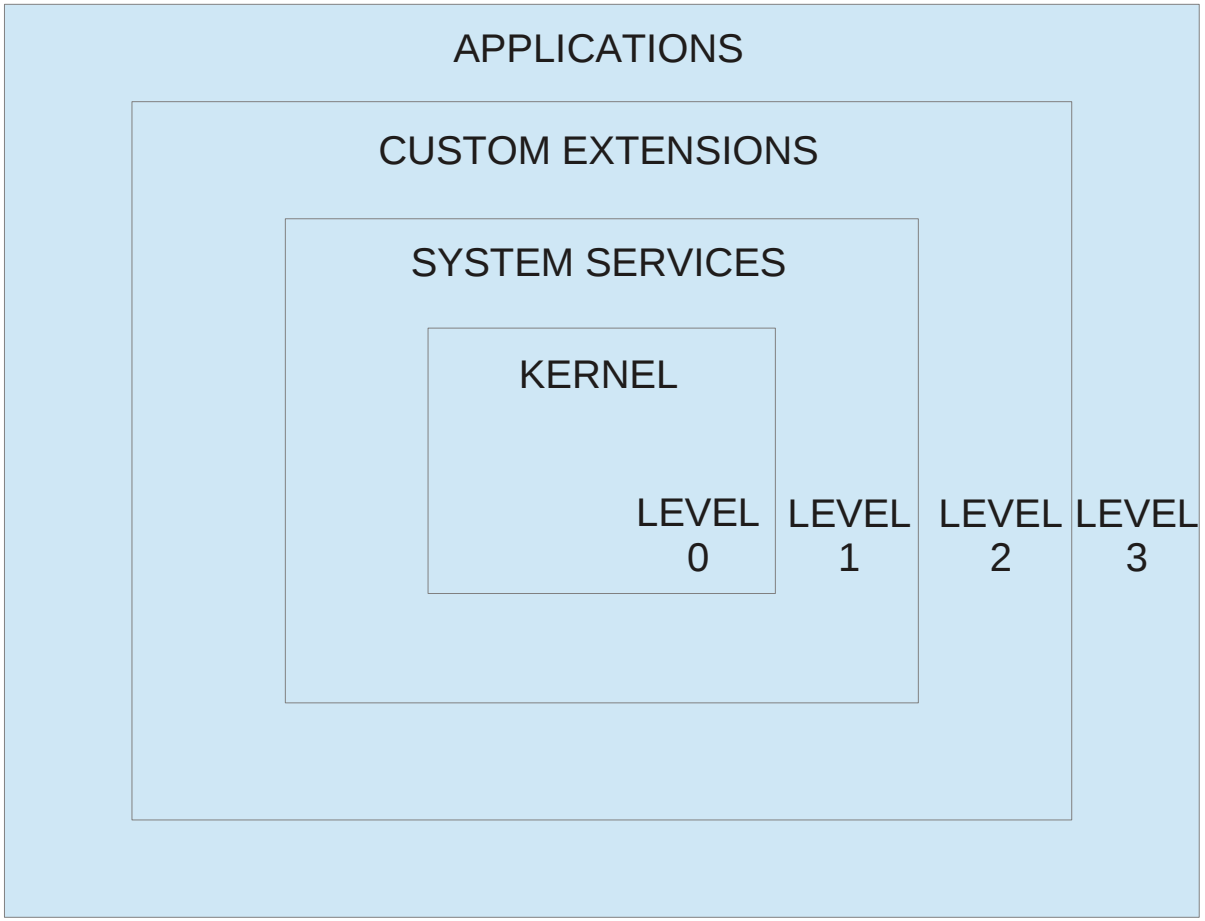
- Registradores Seletores de segmento
  - CS,DS,ES,FS,GS,SS



Linux: Kernel = 00, User = 11

Níveis de privilégio

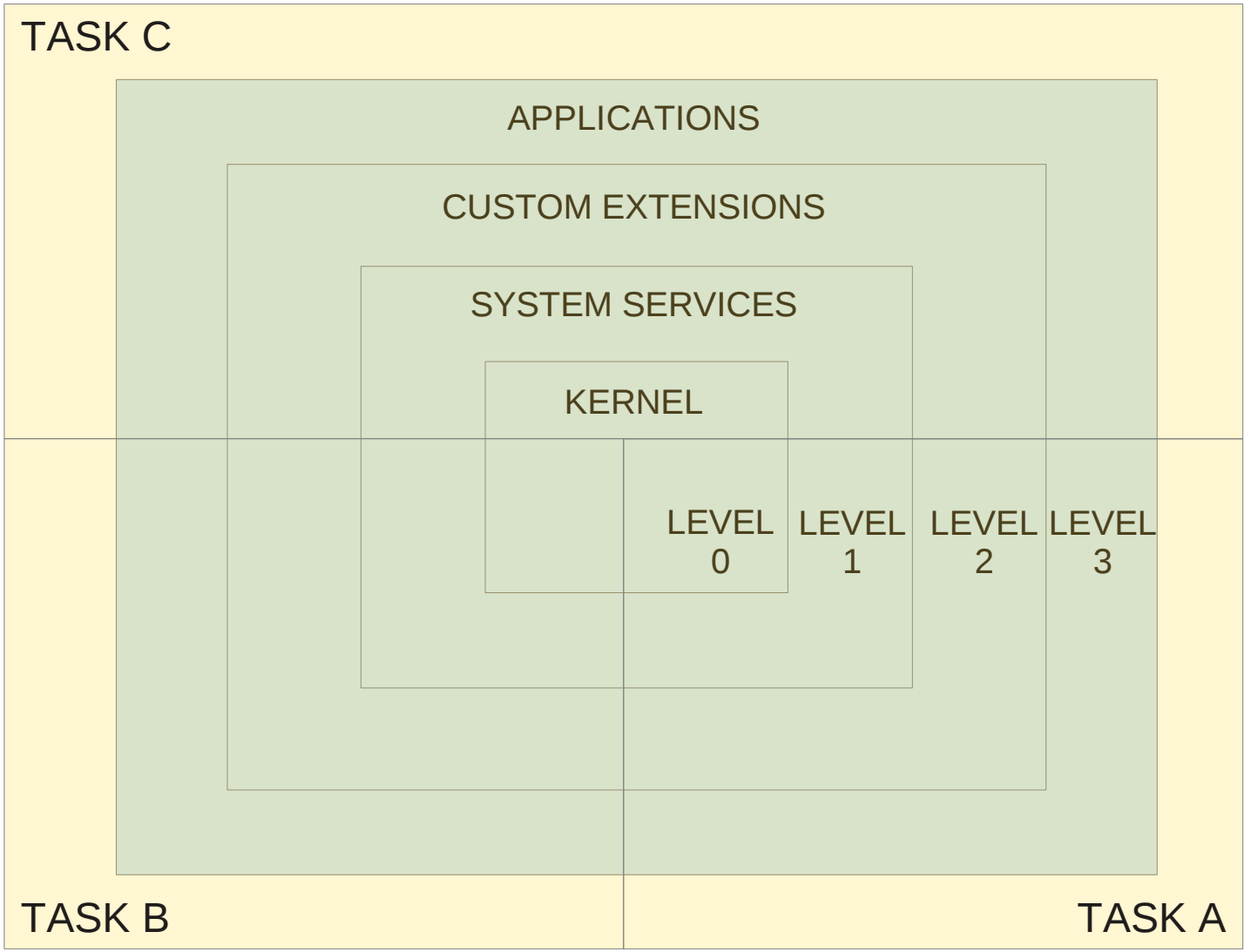
- INTEL 80386 PROGRAMMER'S REFERENCE MANUAL, 1986



- “A one-level system should use privilege level zero”
- “A two-level system should use privilege levels zero and three”

Níveis de privilégio

- INTEL 80386 PROGRAMMER'S REFERENCE MANUAL, 1986



Níveis de privilégio

- AMD64 Architecture Programmer’s Manual Volume 2: System Programming

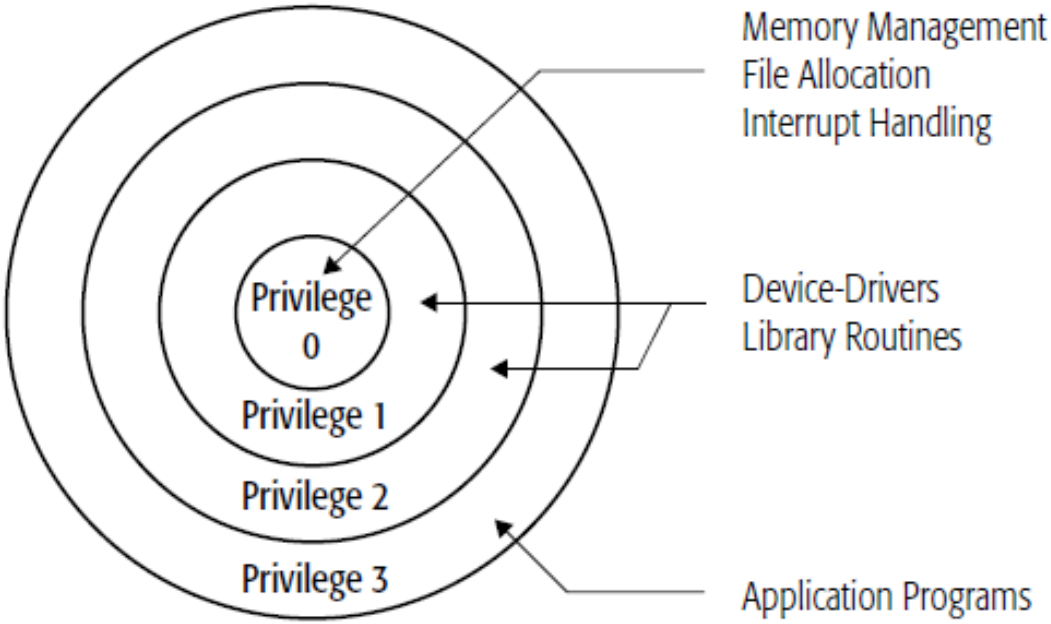
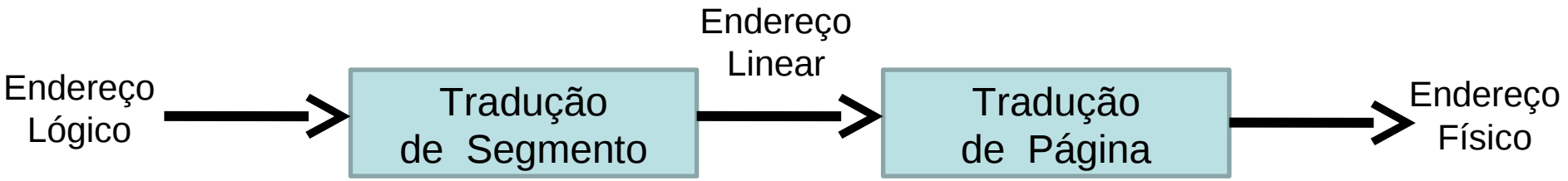


Figure 4-25. Privilege-Level Relationships

Endereçamento de Memória

Adoção de segmentação e paginação (opcional)



Registradores de segmento = Seletor de descritor

Tabelas de descritores

- **Global descriptor table** (Tabela única disponível para todas tarefas, geralmente com dados e código adotados pelo O.S)
- **Task descriptor table** (Usualmente, 1 para cada programa)
- **Interrupt descriptor table** (Adotado para processamento de interrupção)

Endereçamento de Memória

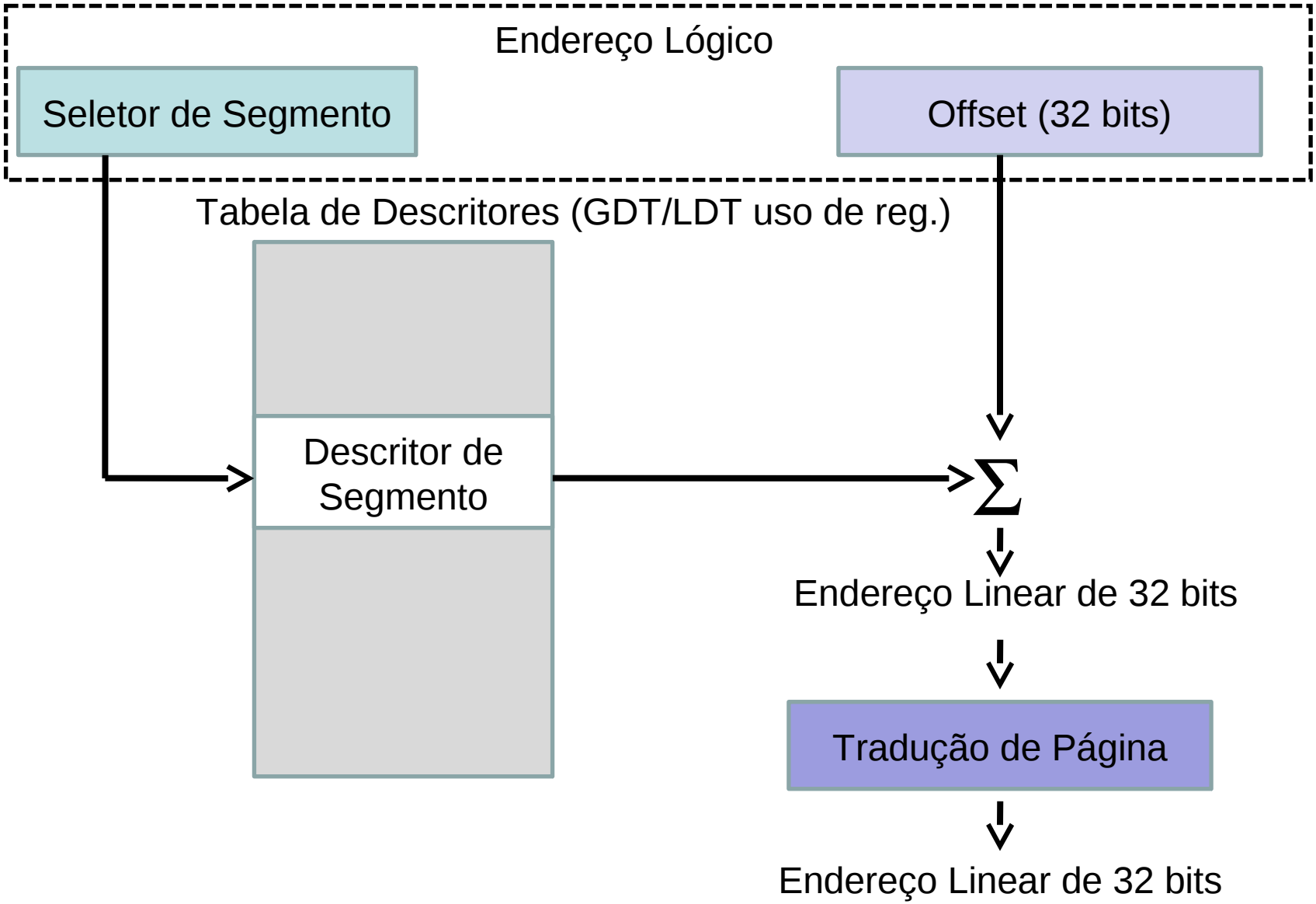
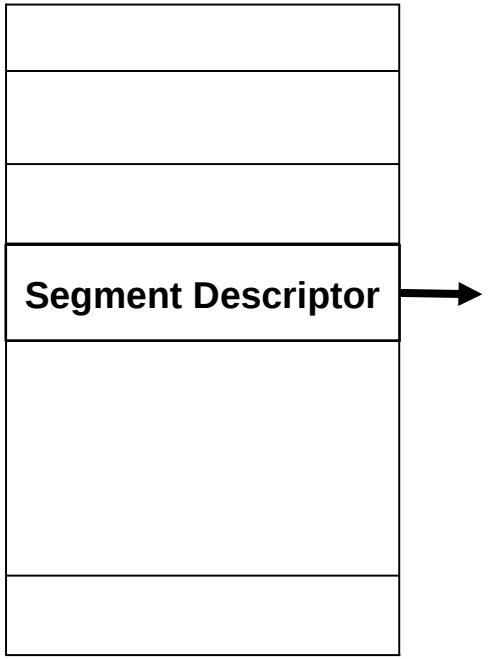


Tabela de descritores



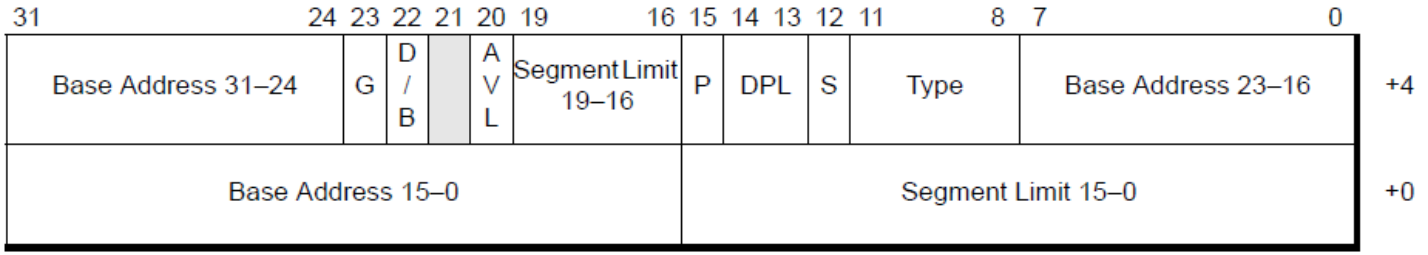


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Legacy Mode = Modo protegido 32 bits

Descritor de segmento (genérico)

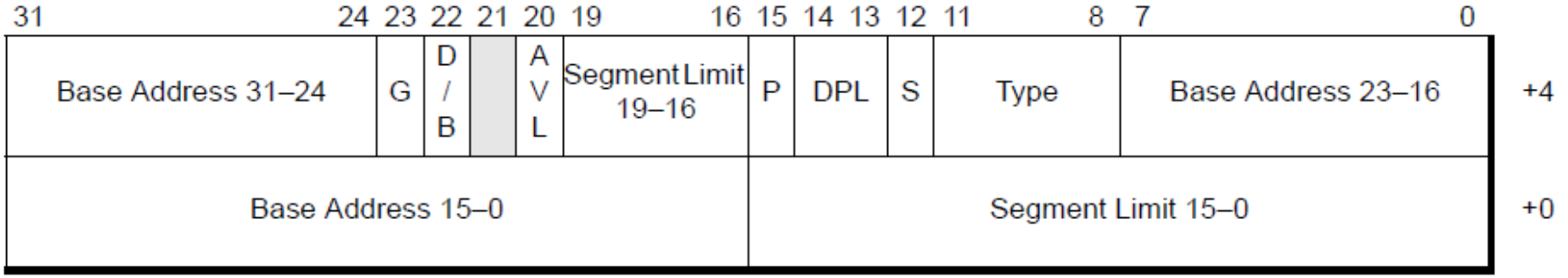


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Segment limit:

- Especifica o tamanho do segmento. O processador “junta os dois pedaços” do campo “segment limit” para formar um valor de 20 bits. O processador interpreta o limite de segmento de uma das seguintes maneiras, dependendo do valor do bit (flag) G (granularidade – granularity) :
  - Se G é 0, então o tamanho do segmento varia de 1 byte a 1MByte, em incrementos de 1 byte.
  - Se G é 1, então o tamanho do segmento vai de 4 KBytes a 4 Gbytes, em incrementos de 4KBytes

Descritor de segmento (genérico)

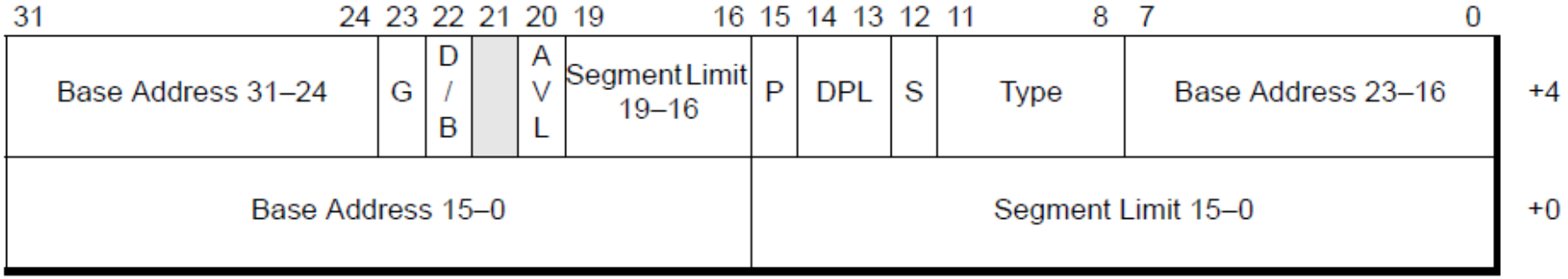


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Base Address:

- Define a localização do byte 0 do segmento dentro do espaço de endereçamento (4GB). O processador “junta os três pedaços” do campo “base address” para formar o endereço de 32 bits.
- Deve ser alinhado a 16bytes (múltiplo de 16 bytes), para melhor performance

Descritor de segmento (genérico)

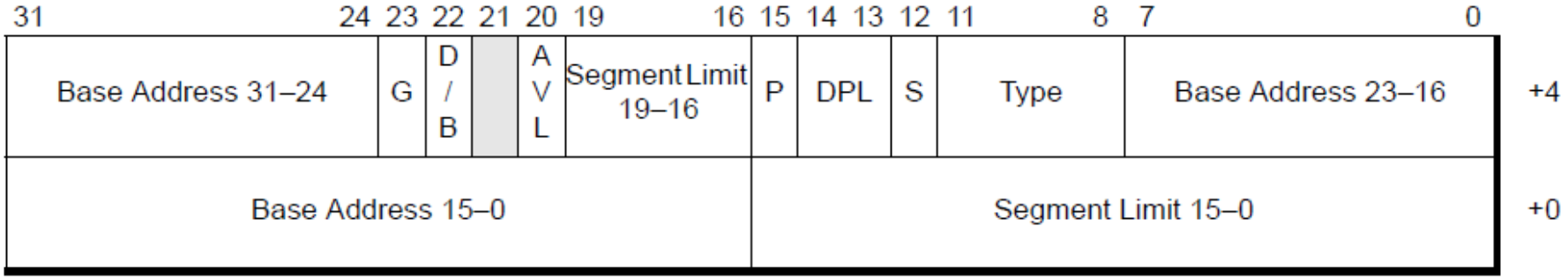


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Type:

- Sua interpretação depende do tipo do segmento (será visto posteriormente)

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) (detalhes a seguir)

DPL:

- Especifica o nível de privilégio do segmento. (de 0 a 3, sendo 0 o mais privilegiado). Usado para controlar o acesso ao segmento.

Descritor de segmento (genérico)

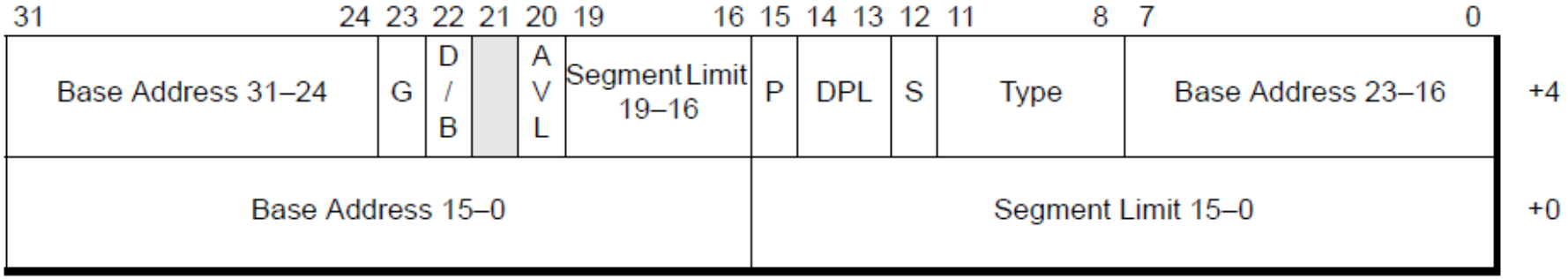


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Flag P (segmento presente):

- Indica se o segmento está presente em memória (P = 1) ou não está presente (P=0). Se P = 0 o processador gera uma exceção “segment not present” quando um seletor de segmento que aponta para esse segmento é carregado num registrador de segmento. O código de gerenciamento de memória do sistema operacional pode usar esse flag para controlar que segmentos estão carregados na memória física a cada instante. (controle adicional ao mecanismo de Paging).

Descritor de segmento (genérico)

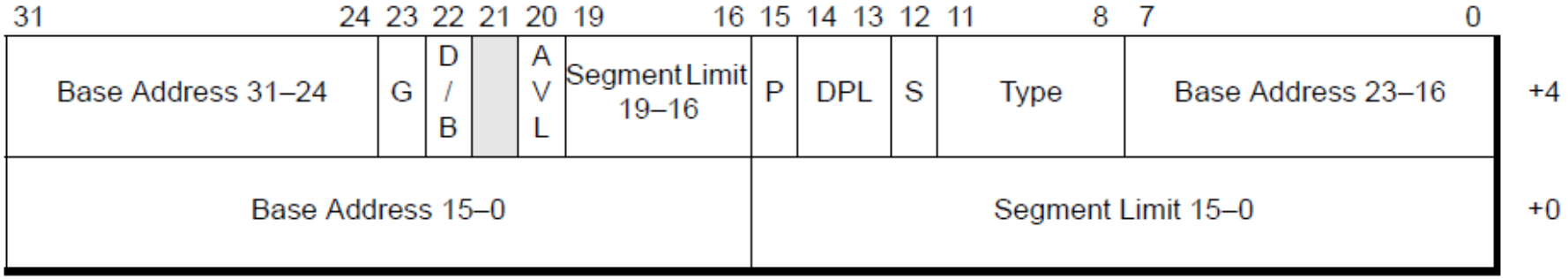


Figure 4-13. Generic Segment Descriptor—Legacy Mode

Flag D/B (default operation size/default stack pointer size and/or upper bound flag):

- Seu papel depende do tipo do segmento (será visto posteriormente)

Descritores para segmentos de código/dados

- Quando o flag S é 1, o descritor é para um segmento de código ou de dados.
- Neste caso, o bit mais alto (11) do campo tipo determina se o segmento é de dados (bit 0) ou código (bit 1)
- **Segmentos de dados:**
    - Os bits 8, 9, 10 do campo tipo são interpretados como (A)cessed, (W)rite-enable, e (E)xpansion-direction.
    - Podem ser somente leitura ou leitura/escrita, dependendo do bit (W)
    - Segmentos de pilha são segmentos de dados que obrigatoriamente são leitura/escrita. Carregar o registrador SS com um seletor de segmento que aponte para um segmento somente leitura gera uma exceção (general-protection). Se o tamanho de um segmento de pilha precisa ser mudado dinamicamente, o segmento de pilha pode ser do tipo “expande para baixo” (E = 1). Nesse caso, ao mudar dinamicamente o limite do segmento faz-se com que espaço de pilha seja adicionado na base da pilha.
    - O bit (A) indica se o segmento foi acessado desde a última vez que o sistema operacional fez o bit = 0. O processador faz o bit = 1 sempre que ele carrega um seletor de segmento para o segmento num registrador de segmento. O bit permanece = 1 até que seja explicitamente setado para 0. Isso pode ser usado para gerenciamento de memória e para debugging.

Descritores para segmentos de código/dados

- **Segmentos de código:**
  - Os bits 8, 9, 10 do campo tipo são interpretados como (A)cessed, (R)ead-enable, e (C)onforming.
  - Segmentos de código podem ser “execute-only” ou “execute/read”, dependendo do bit (R). Um segmento execute/read pode ser usado quando constantes ou outros dados estáticos foram colocados junto com instruções em uma ROM. Nesse caso, dados podem ser lidos do segmento de código usando umas instrução com um prefixo de override com CS ou carregando um seletor de segmento para um registro de segmento de dados (DS, ES, FS ou GS). Em modo protegido, segmentos de código não são writable.
  - Segmentos de código podem ser conforming ou nonconforming. Uma transferência de execução para um segmento conforming de maior privilégio permite que a execução continue no nível atual de privilégio. Uma transferência de execução para um segmento nonconforming de um nível de proteção diferente gera uma exceção (general-protection), a menos que um call-gate ou task-gate seja usado.
  - Todos segmentos de dados são nonconforming (não podem ser acessados por programas/procedimentos de menor privilégio). No entanto, segmentos de dados podem ser acessados por programas/procedimentos de maior privilégio

RESUMO: tipos de segmentos de dados/código

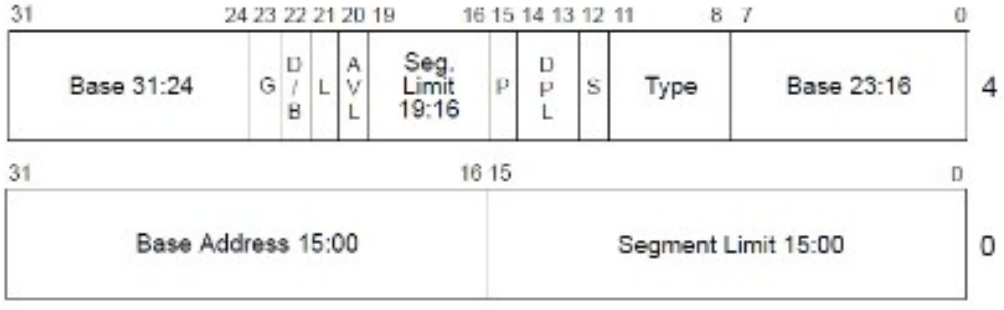
Table 3-1. Code- and Data-Segment Types

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
		C	R	A		
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, conforming
15	1	1	1	1	Code	Execute/Read, conforming, accessed

Exercício de fixação (do EE de 2012.1)

Cansado de esperar vários minutos para o seu computador inicializar a cada vez que o liga, Joe Bitwise, último representante de uma tradicional família de programadores, decide criar o seu próprio sistema operacional para processadores da família X86 (Pentium e superiores), escrito em assembly otimizado manualmente. Ao estudar os processadores desta família, ele descobre que poderia usar o modo real ou o modo protegido (32 bits, Joe quer que seu OS rode mesmo nos seus computadores mais antigos que não possuem modo 64 bits). Usando o NASM, Joe define um descritor da seguinte maneira:

```
dw 0xFFFF
dw 0
db 0
db 0x9A
db 0x5C
db 0
```



L — 64-bit code segment (IA-32e mode only)
AVL — Available for use by system software
BASE — Segment base address
D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
DPL — Descriptor privilege level
G — Granularity(0 = in bytes unit; 1 in 4 Kbyte unit)
LIMIT — Segment Limit
P — Segment present
S — Descriptor type (0 = system; 1 = code or data)
TYPE — Segment type

Figure 3-8. Segment Descriptor

Perguntas (responda, justificando):
1) Trata-se de um descritor de sistema?
2) Qual o endereço base do segmento descrito?
3) Qual o tamanho do segmento descrito?

Continua...