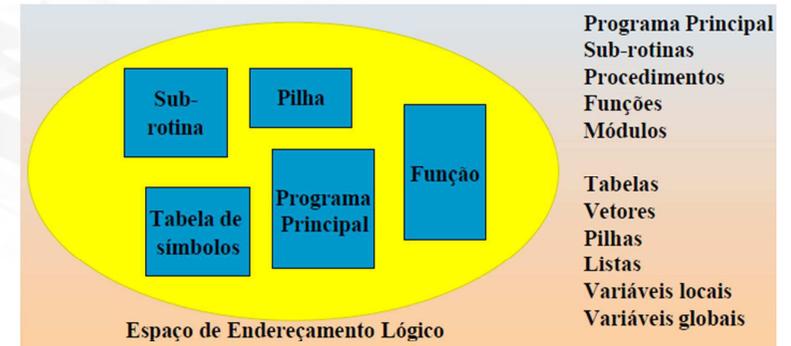


# Gerência de Memória

## Segmentação

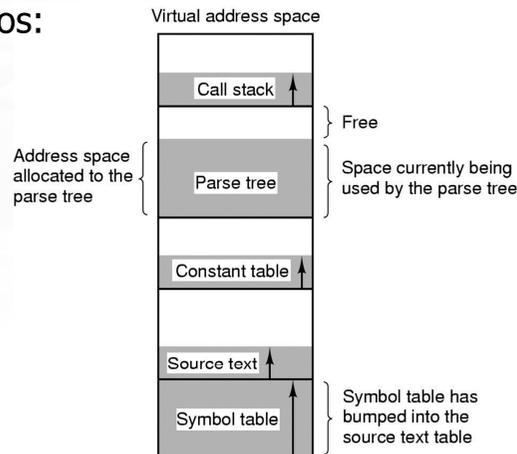
## Qual a visão que o usuário tem da memória?



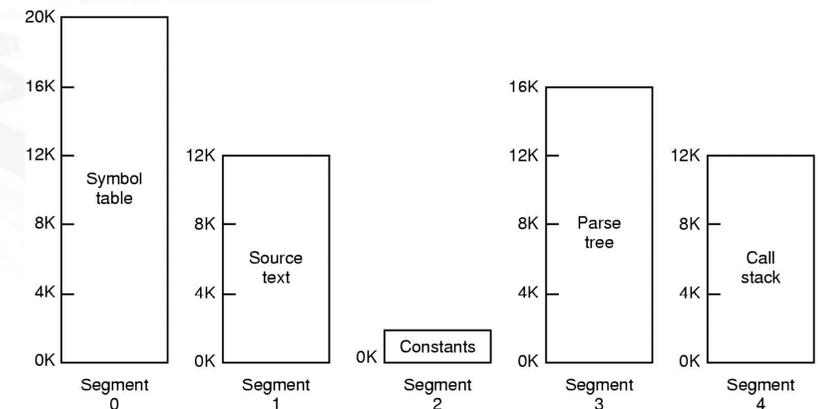
- Com a paginação ocorre a separação entre a visão da memória pelo usuário e a memória física

## Memória Segmentada (1)

- Programas são normalmente separados em módulos: unidade lógica



## Memória Segmentada (2)



Permite que cada tabela cresça ou encolha, independentemente!

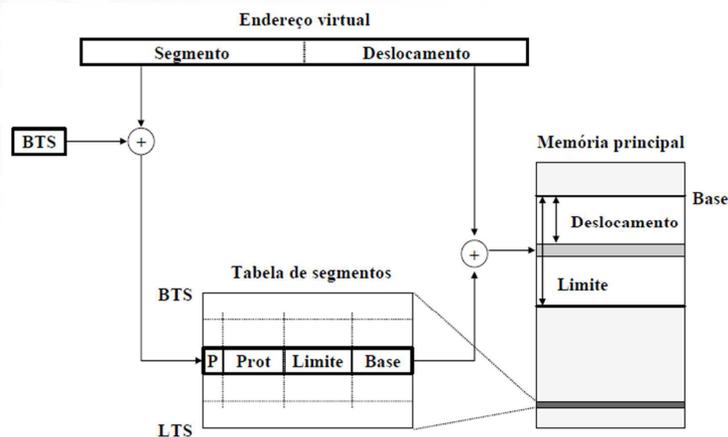
## Memória virtual: Segmentação

- Divisão dos programas em segmentos lógicos que refletem a sua estrutura funcional
  - rotinas, módulos, código, dados, pilha, etc.
- O objetivo da segmentação é dividir programas em seções para que o S.O. possa relocá-los mais facilmente na memória
- Programa é dividido em segmentos, que são blocos de endereços
  - O espaço de endereçamento virtual é linear **em cada segmento**
  - Segmentos de um programa não precisam ser do mesmo tamanho
  - A dimensão dos segmentos é limitada pela arquitetura
  - O compilador constrói automaticamente os segmentos
- Usuário tem controle
  - O programador pode ter que se preocupar com a gestão de memória quando escreve um programa

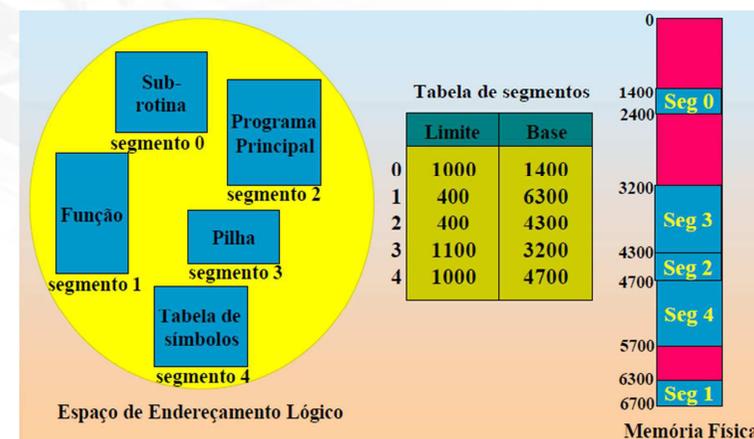
## Segmentação: Endereçamento (1)

- Endereço dividido em 2 partes
  - número do segmento (ou base)
  - deslocamento (ou offset)
- Segmento e deslocamento devem ser somados, e não concatenados
- Uma tabela de segmento para cada processo ativo
- Registrador especial contém endereço inicial da tabela de segmento

## Segmentação: Endereçamento (2)



## Segmentação: Endereçamento (3)



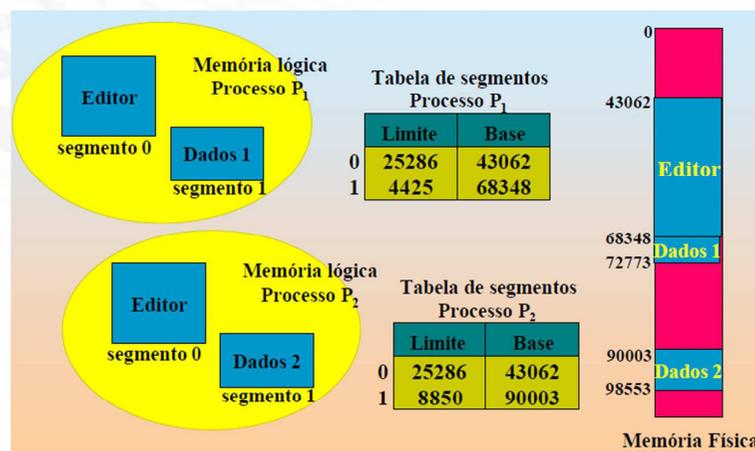
## Segmentação: Endereçamento (4)

- Tabela de segmento contém
  - comprimento do segmento
    - armazenado na tabela de segmento para evitar que programa acesse erradamente posições fora do segmento
  - bits de proteção de memória
  - bits para o algoritmo de substituição
- Proteção de memória: segmento pode ser
  - read-only
  - execute-only
  - system-only

## Segmentação: Vantagens

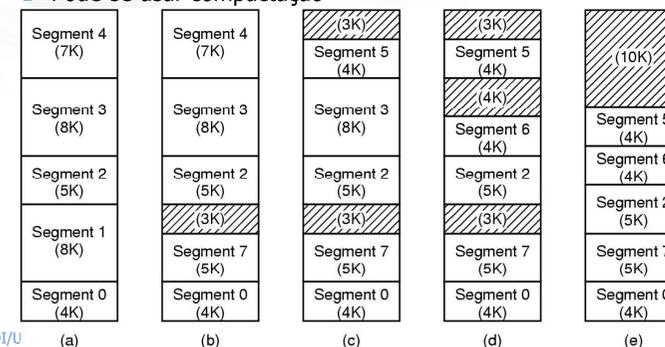
- Compartilhamento de memória entre processos:
  - Basta colocar nas tabelas de segmentos dos processos em questão o endereço real do segmento a compartilhar
  - Os endereços virtuais usados para acessar o segmento com partilhado podem ser diferentes nos vários processos
  - A proteção de um segmento compartilhado é definida para cada processo através da respectiva tabela de segmentos

## Compartilhamento de Segmentos



## Segmentação: Problemas

- Algoritmo de substituição: mais complexo do que em paginação devido ao tamanho variável dos segmentos (eg. First fit, best fit,..)
- Fragmentação externa
  - Segmentos de tamanhos variáveis
  - Pode-se usar compactação



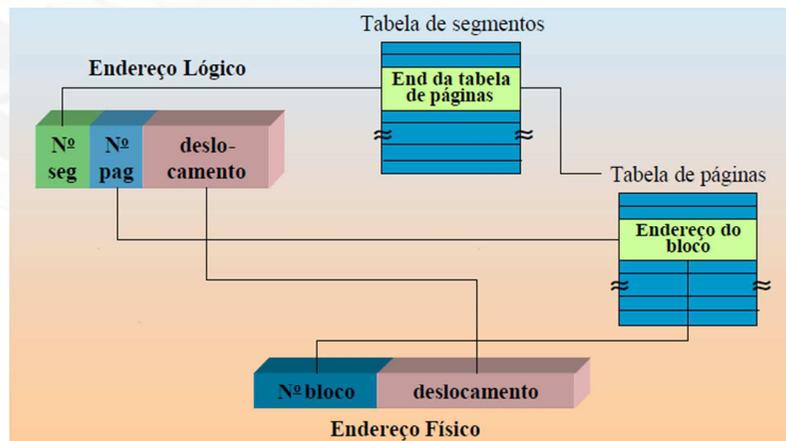
## Paginação x Segmentação

Consideração	Paginação	Segmentação
O programador precisa estar ciente de que essa técnica está sendo usada?	Não	Sim
Quantos espaços de endereçamentos lineares existem?	Um	Muitos
O espaço de endereçamento total pode exceder o tamanho da memória física?	Sim	Sim
Os procedimentos e os dados podem ser diferenciados e protegidos separadamente?	Não	Sim
As tabelas com tamanhos variáveis podem ser acomodadas facilmente?	Não	Sim
O compartilhamento de procedimentos entre usuários é facilitado?	Não	Sim
Por que essa técnica foi inventada?	Para fornecer um grande espaço de endereçamento linear sem a necessidade de comprar mais memória física	Para permitir que programas e dados sejam quebrados em espaços de endereçamento logicamente independentes e para auxiliar o compartilhamento e a proteção

## Segmentação com Paginação (1)

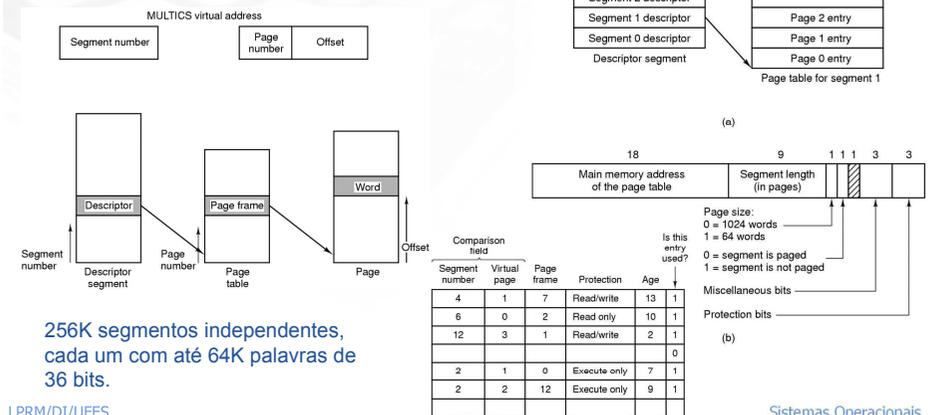
- A paginação é a solução natural para a fragmentação
- Recuperar as vantagens dos dois métodos em relação a fragmentação:
  - Fragmentação interna: paginação apresenta, segmentação não
  - Fragmentação externa: segmentação apresenta, paginação não
- Solução se traduz em paginar segmentos

## Segmentação com Paginação (2)

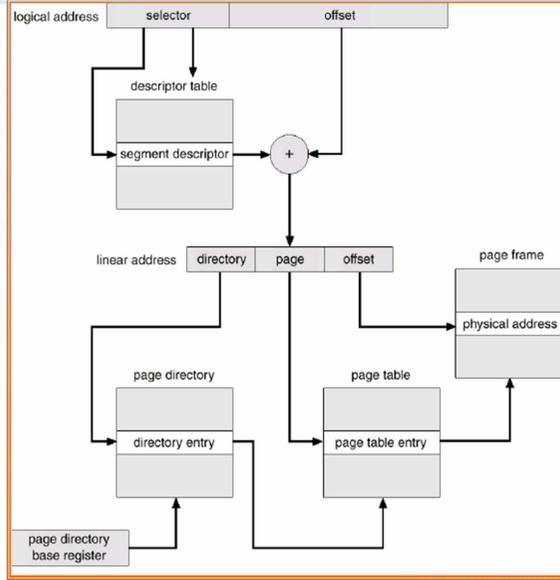


## Segmentação no S.O. MULTICS

(p/ processador Honeywell 6000)

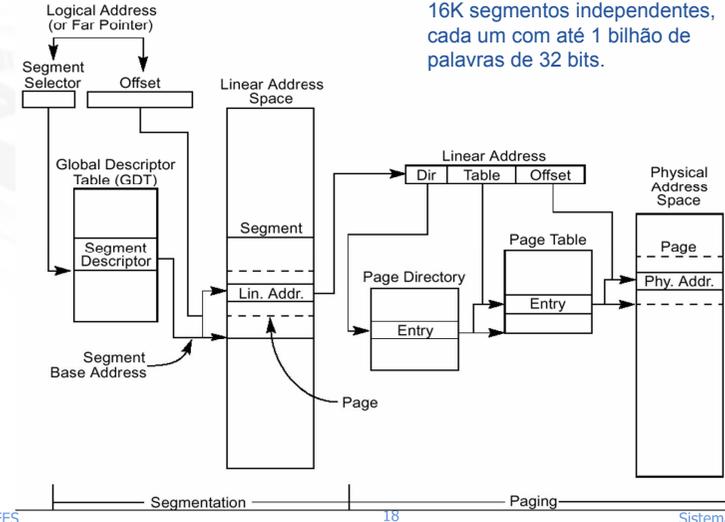


# Arquitetura de memória do i386



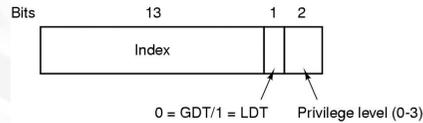
# Arquitetura de Memória do Pentium (1)

16K segmentos independentes, cada um com até 1 bilhão de palavras de 32 bits.

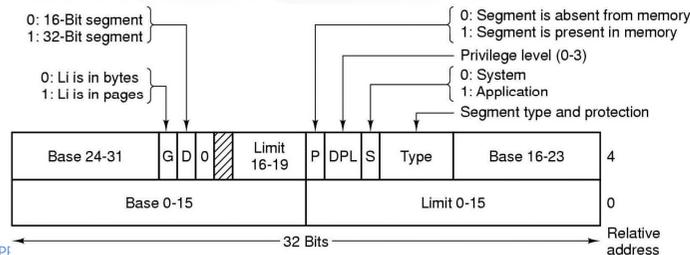


# Arquitetura de Memória do Pentium (2)

## Formato do seletor

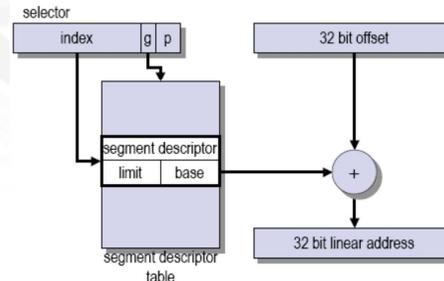


## Entrada na Tabela de Segmentos (2 x 4 bytes)

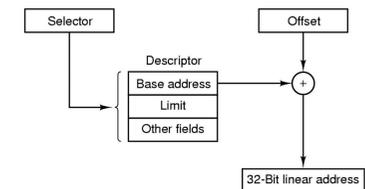


# Arquitetura de Memória do Pentium (3)

Step 1: Use the Selector to convert the 32 bit virtual offset address to a 32 bit linear address.



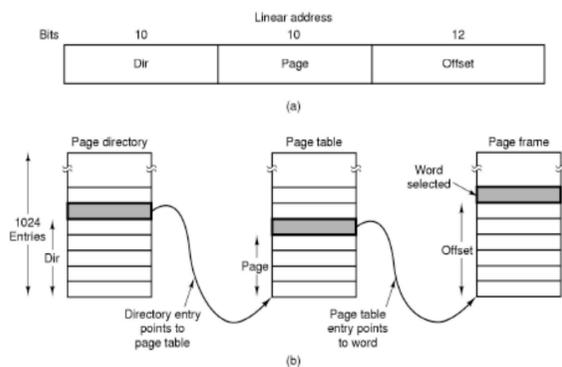
Transformação do par (segmento, desloc.) em um endereço linear



## Arquitetura de Memória do Pentium (4)

Transformação do par (segmento, desloc.)  
em um endereço linear (cont.)

Step 2: Convert the 32 bit linear address to a physical address using a two-stage page table.



## Referências

- **A.S. Tanenbaum, "Sistemas Operacionais Modernos", 3a. Edição, Editora Prentice-Hall, 2010.**
  - Seção 3.7
- **Silberschatz, P. Baer Galvin, e G. Gagne "Fundamentos de Sistemas Operacionais", 6a. Edição, Elsevier Editora / Campus, 2004.**
  - Seções 9.5 e 9.6
- Deitel H. M.; Deitel P. J.; Choffnes D. R.; "Sistemas Operacionais", 3ª. Edição, Editora Prentice-Hall, 2005
  - Seções 10.5 a 10.7