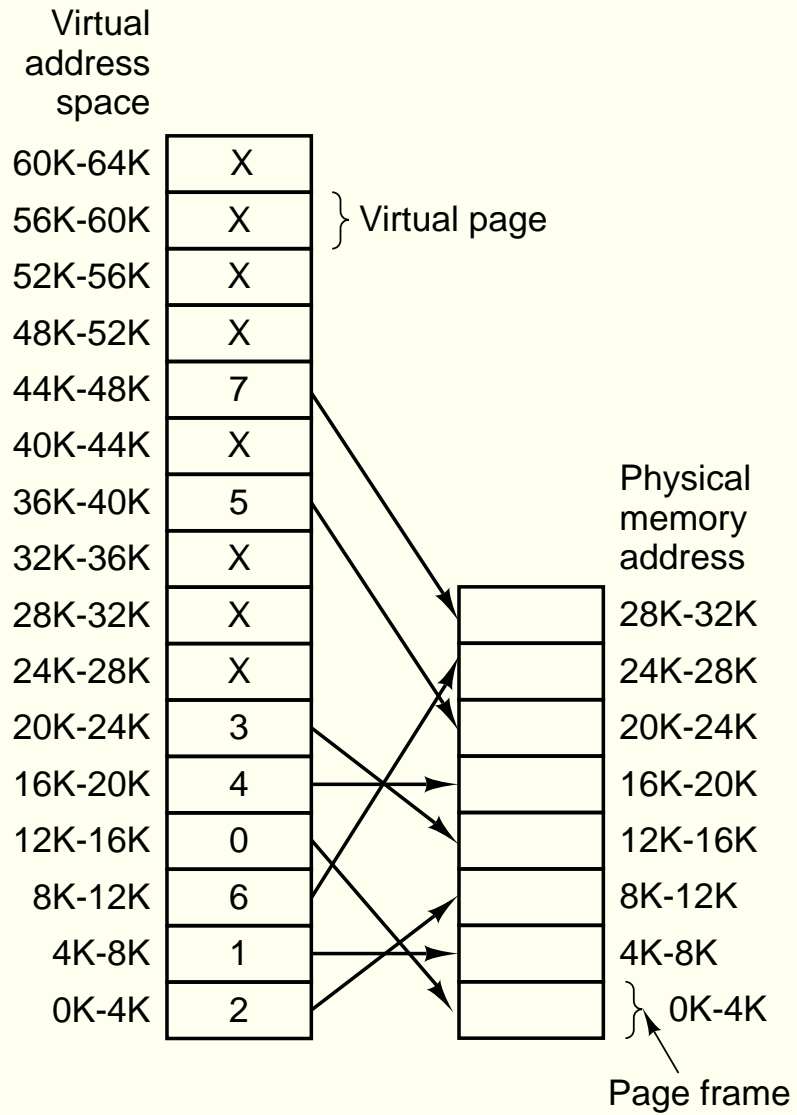


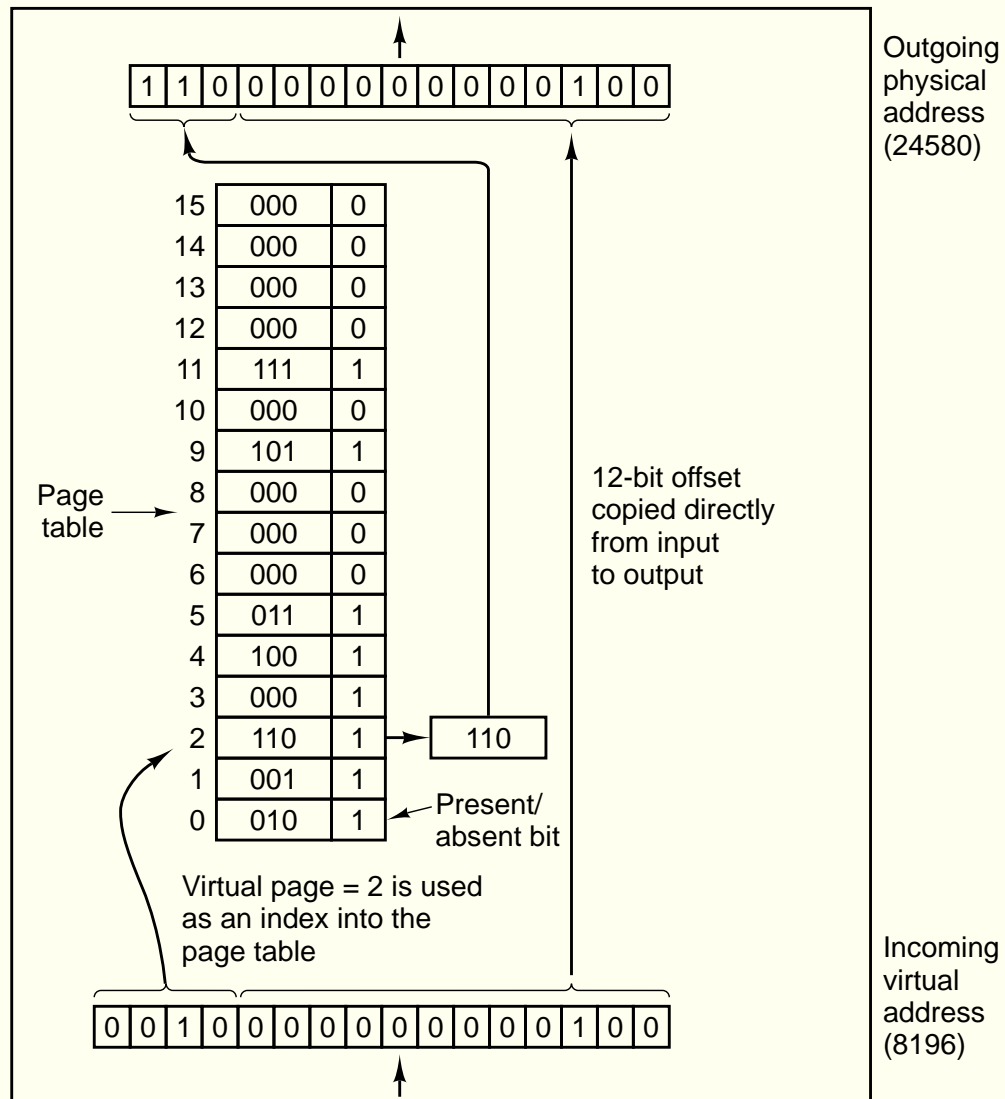
**MC514–Sistemas Operacionais: Teoria e Prática**  
1s2008

## **Gerenciamento de Memória - 2**

# Paginação



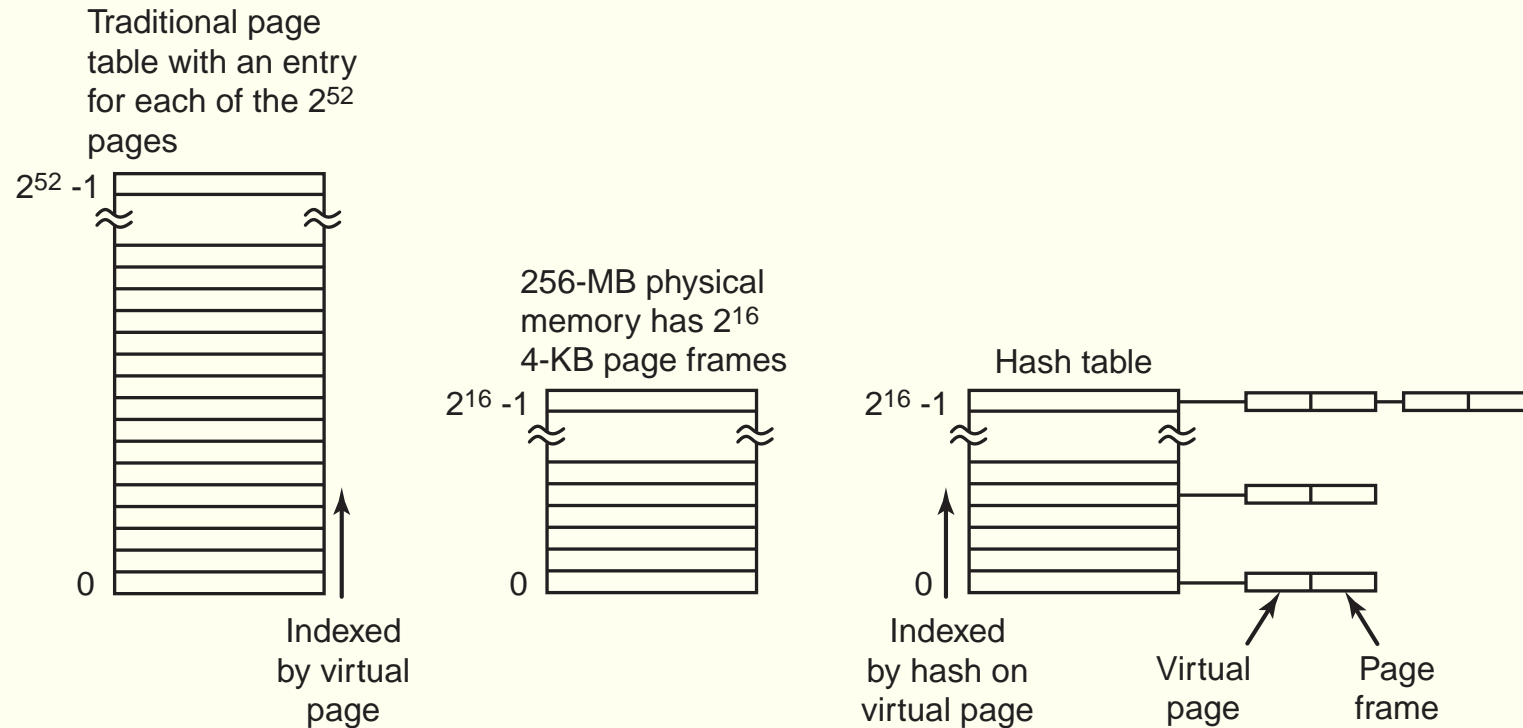
# Mapeamento dos endereços



# Paginação - Exemplo

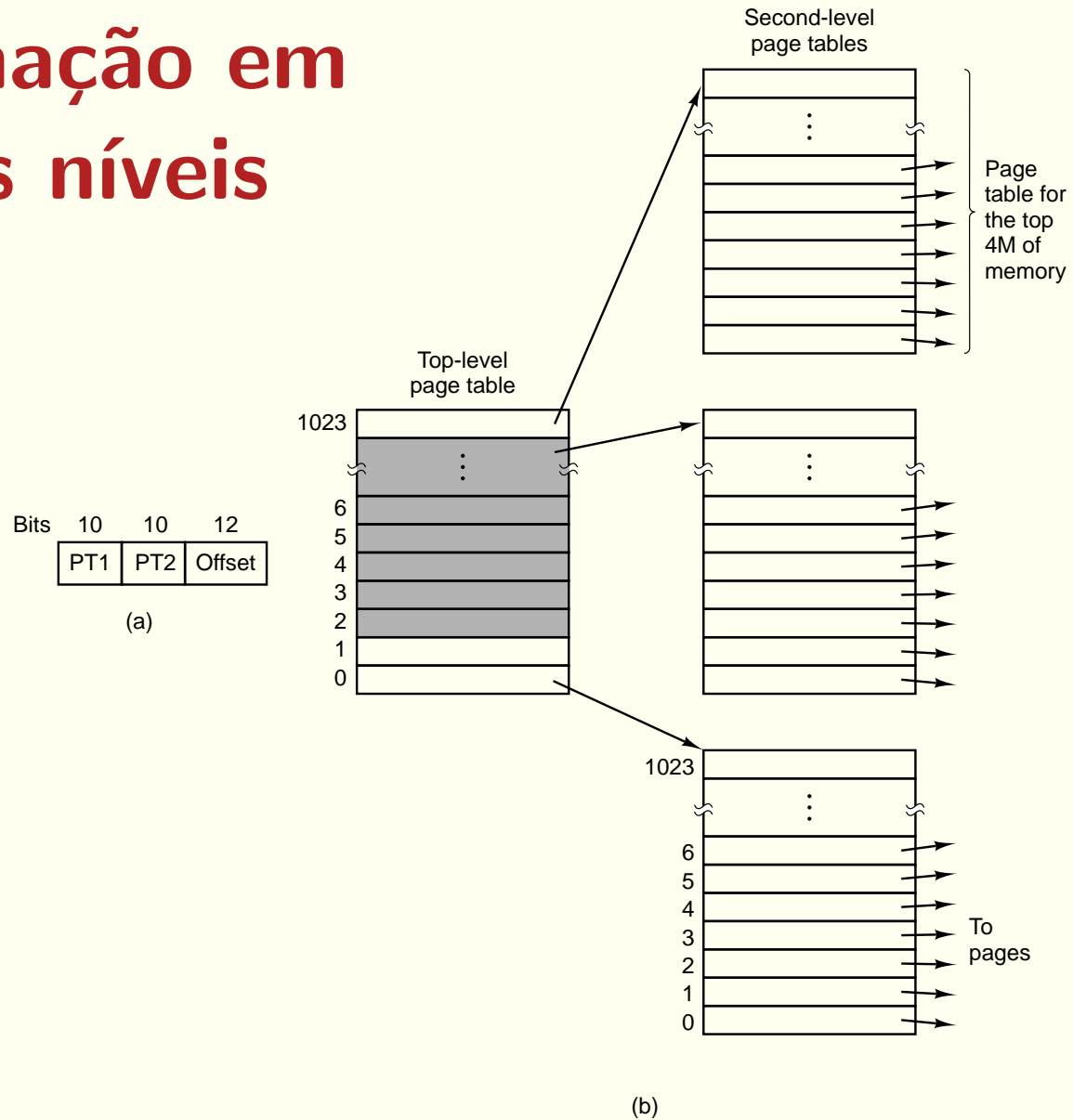
- 32 bits de endereço
- páginas de 4k
- $2^{20}$  entradas na tabela?  
e apenas algumas seriam utilizadas...
- Outras opções
  - tabelas invertidas
  - paginação em vários níveis

# Tabela de páginas invertida

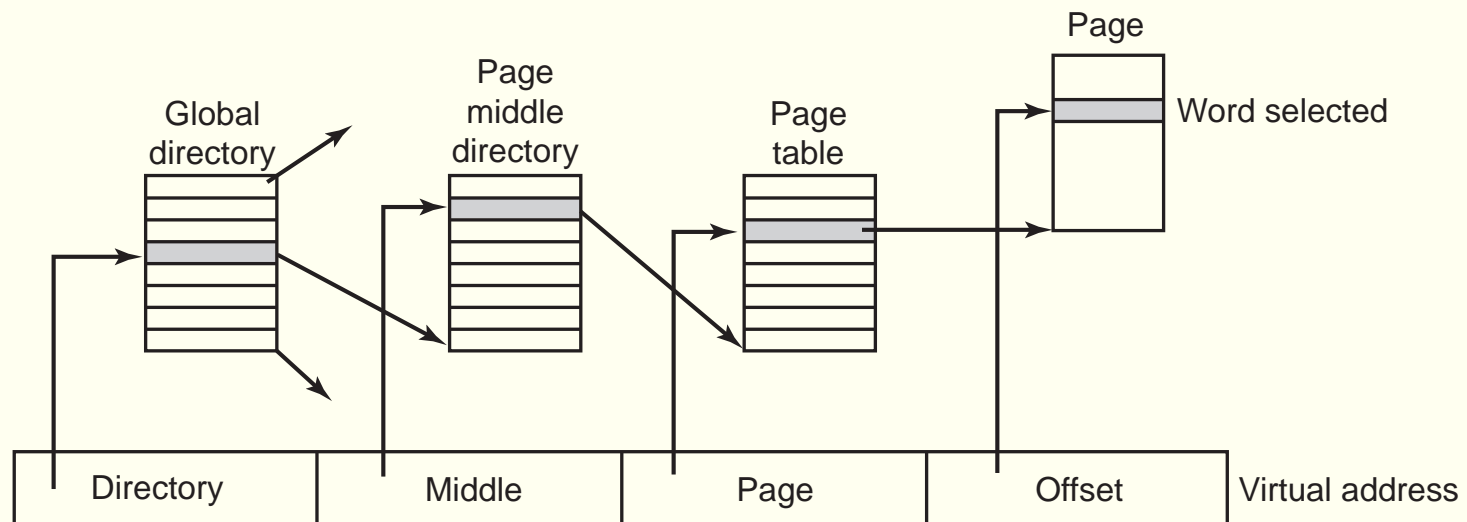


- Uma entrada por página na memória física

# Paginação em vários níveis



# Tabela em três níveis



- Linux?

# Translation Look Aside Buffers (TLBs)

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Vamos analisar os exemplos `page1.c` e `loop_malloc.c`



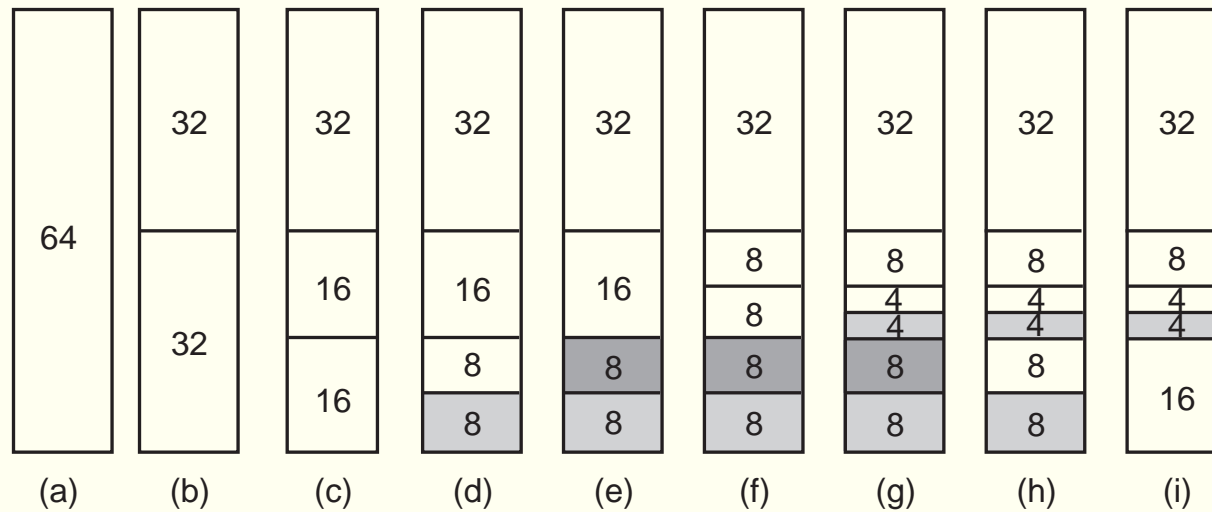
## Política de limpeza

- Gravar as páginas modificadas na última hora pode ser pouco eficiente
- *Paging Daemon*
  - Varre periodicamente a memória
  - Tenta manter um número de *frames* livres
  - kswapd

# Controle da carga

- *Thrashing*
  - O Sistema Operacional só se ocupa das tarefas de paginação e escalonamento
  - Todos os processos precisam de mais memória
  - Swap (como escolher quais processos vão para o disco?)

# Buddy system



- Vamos verificar utilizando sh.c

# Fixação de páginas na memória

- Melhora do desempenho para processos com características especiais

```
int mlock (const void *ADDR, size_t LEN);
```

```
int munlock (const void *ADDR, size_t LEN);
```

- Só para super-usuários? :-)

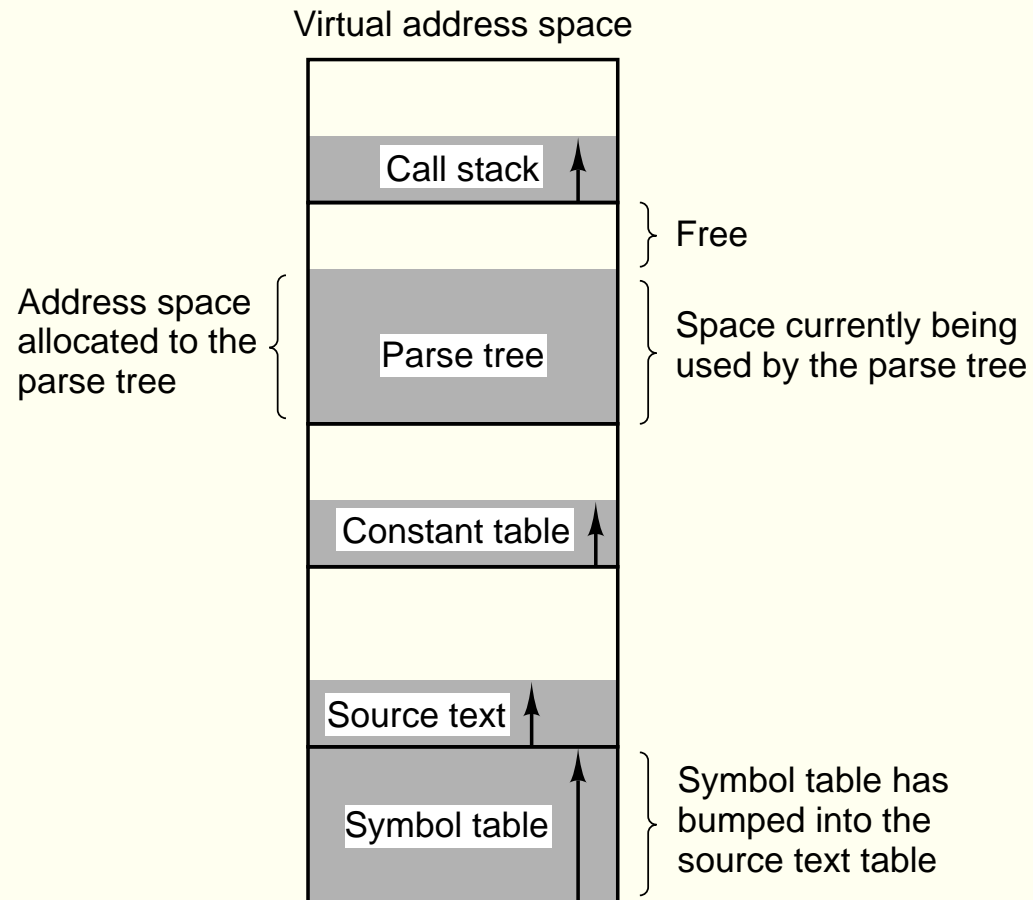
# Mmap

- Um arquivo pode ser mapeado em memória
- Memória compartilhada
- Veja map.c

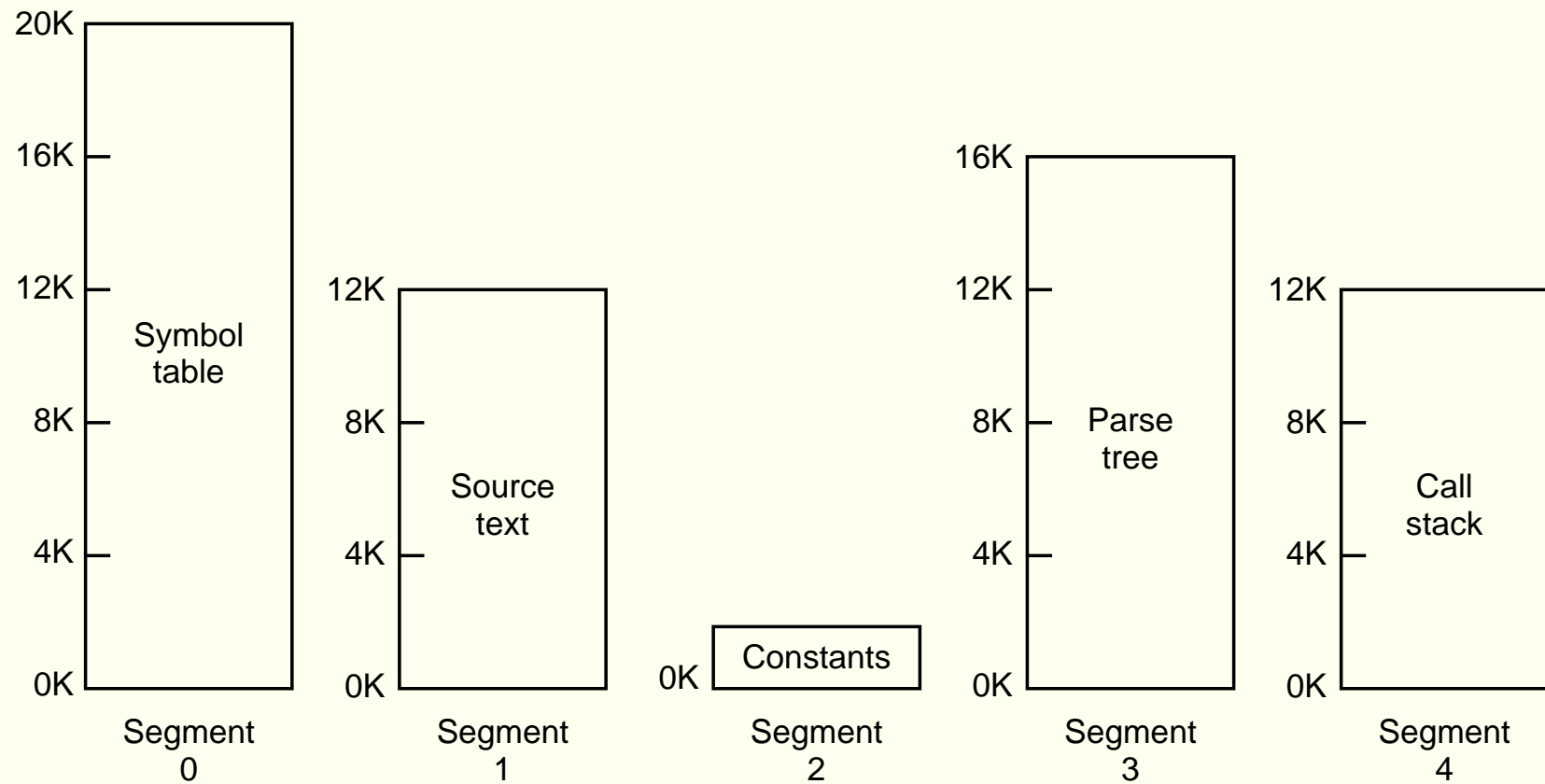
# Segmentação

- Vários espaços de endereçamento
- Maior flexibilidade

# Limites de um espaço único



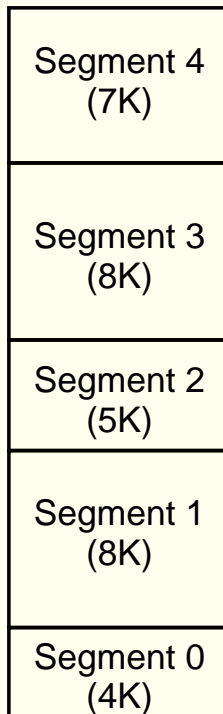
# Segmentação



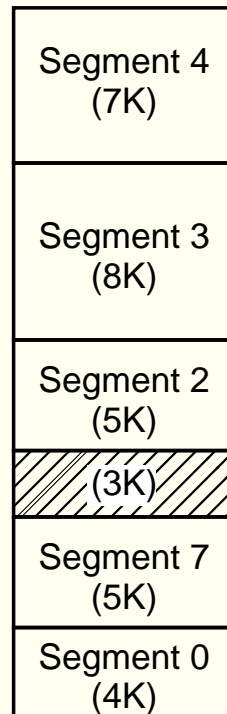


Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

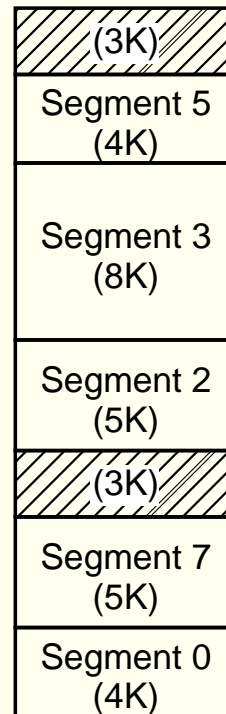
# Segmentação pura



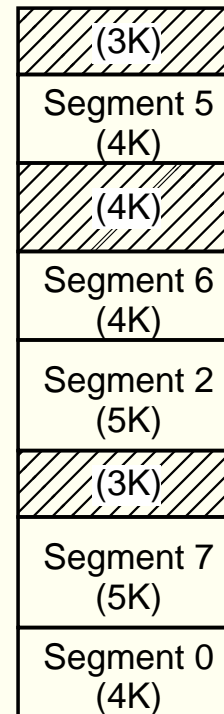
(a)



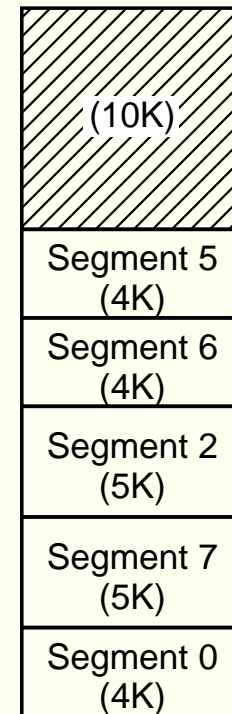
(b)



(c)

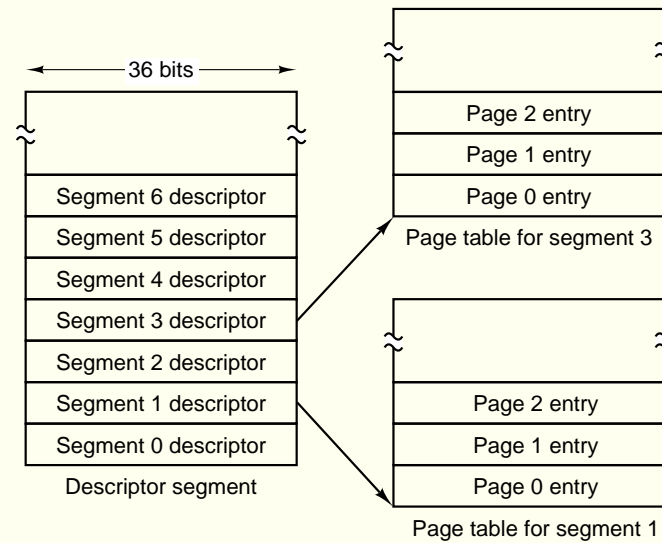


(d)

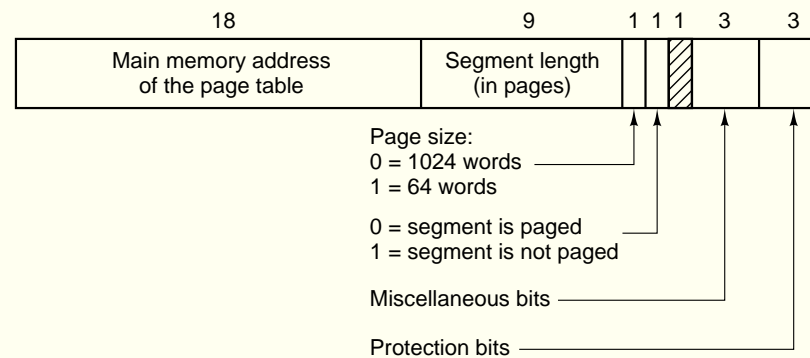


(e)

# MULTICS: segmentação e paginação



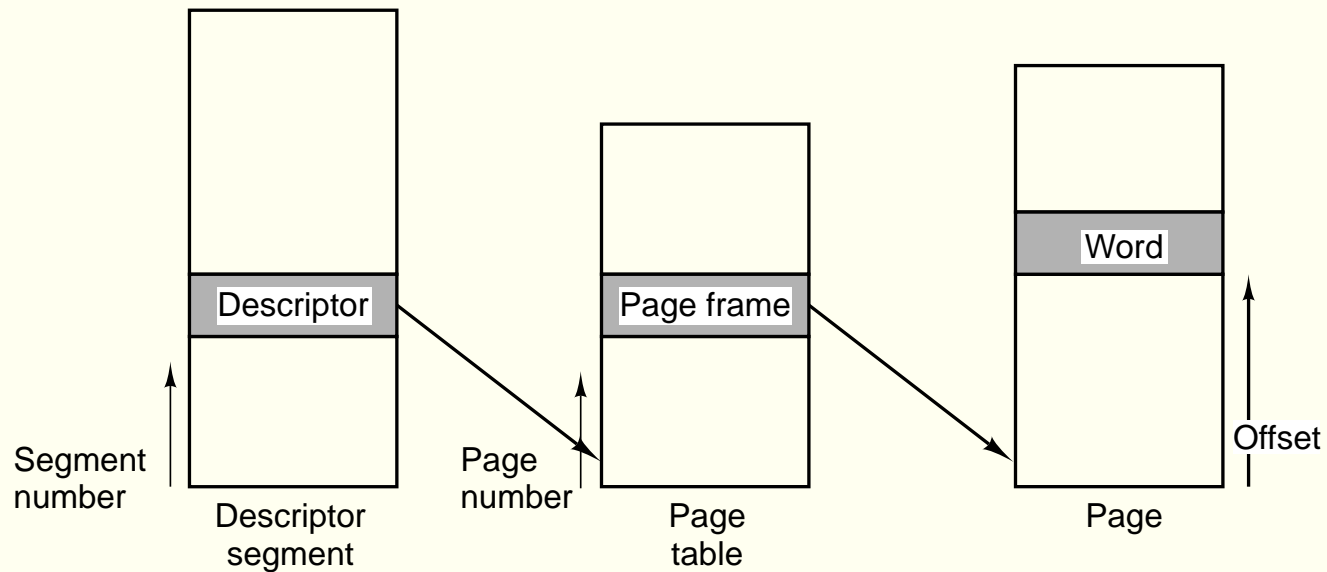
(a)



(b)

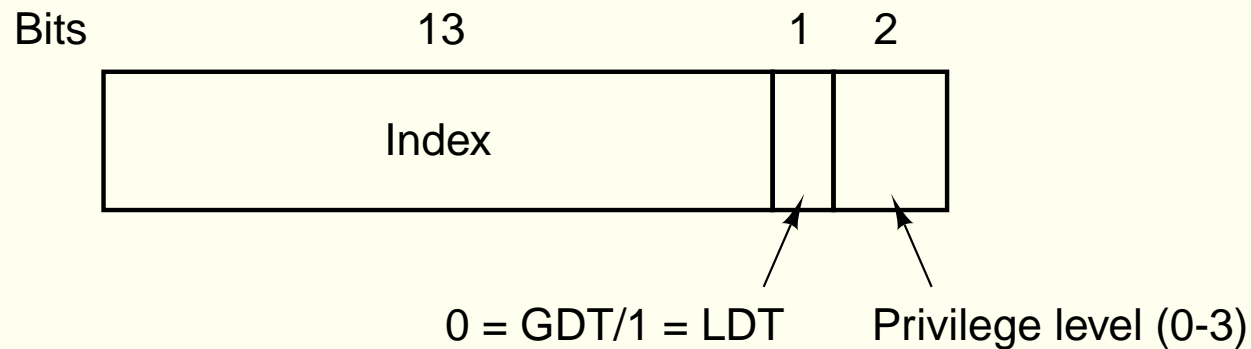
# MULTICS: Conversão de endereços

MULTICS virtual address



# Intel Pentium: segmentação e paginação

- Seletor de segmento

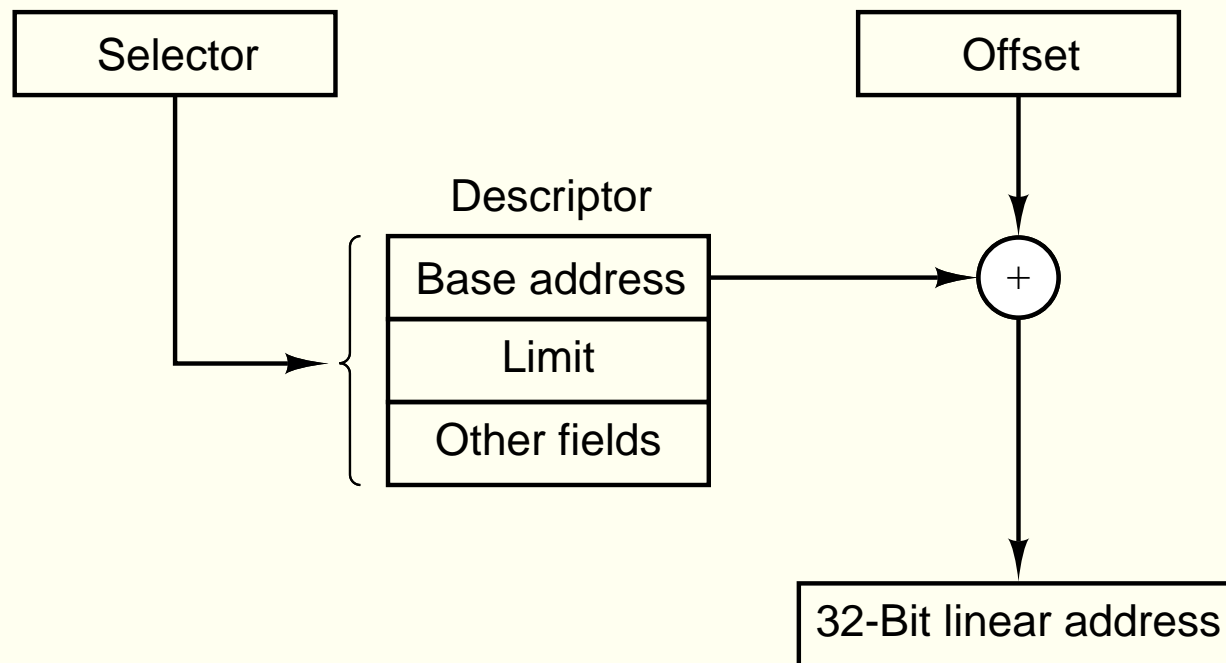


- LDT (Local Descriptor Table)
- GDT (Global Descriptor Table)

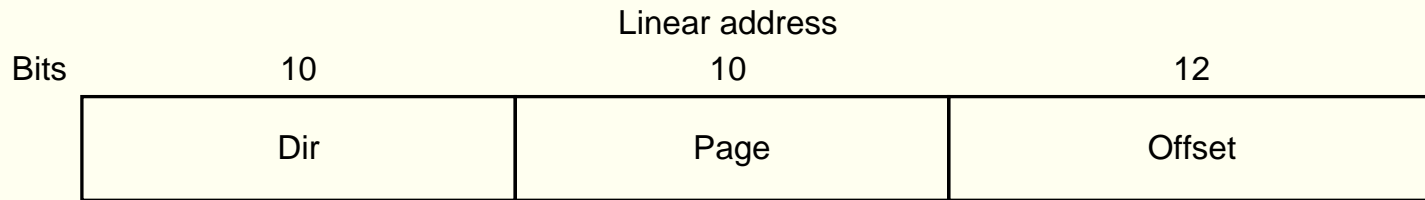
# Registradores de segmento

- CS - Code Segment
- DS - Data Segment
- SS - Stack Segment
- ES, FS, GS - Extra Segments

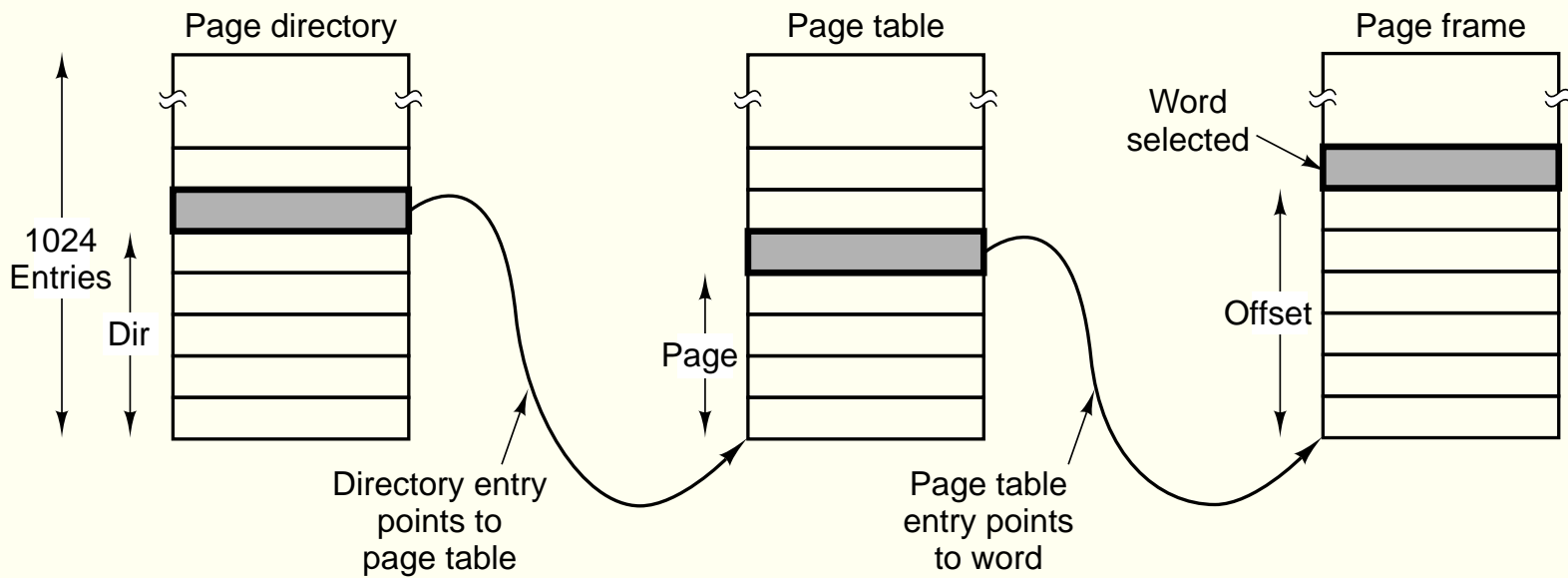
# Conversão de endereços (seletor, deslocamento) → linear



# Conversão de endereços linear → físico



(a)



(b)