Universidade Federal do Rio Grande do Sul
Instituto de Informática

Organização de Computadores

Aula 20

Memória virtual
segunda parte

Revisão Aula Passada

MEMÓRIA VIRTUAL

1. Introdução

técnica que nos permite ver a memória principal como uma cache de grande capacidade de armazenamento, das memórias secundárias.

2. Paginação

mecanismo simples para tradução de endereços virtuais em reais e para gerenciamento do espaço de memória

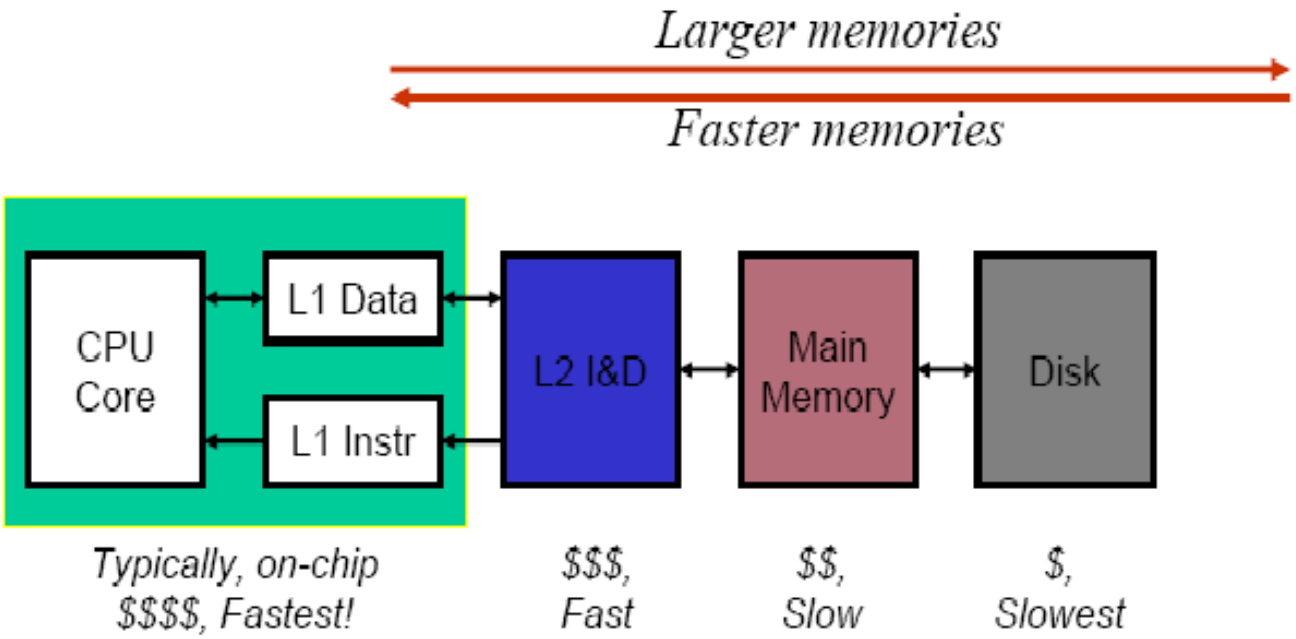
3. Translation Lookaside Buffer TLB

traduz endereços virtuais para endereços reais, podendo ser considerado como uma "cache" da MMTT

4. Mecanismos de translação de endereços

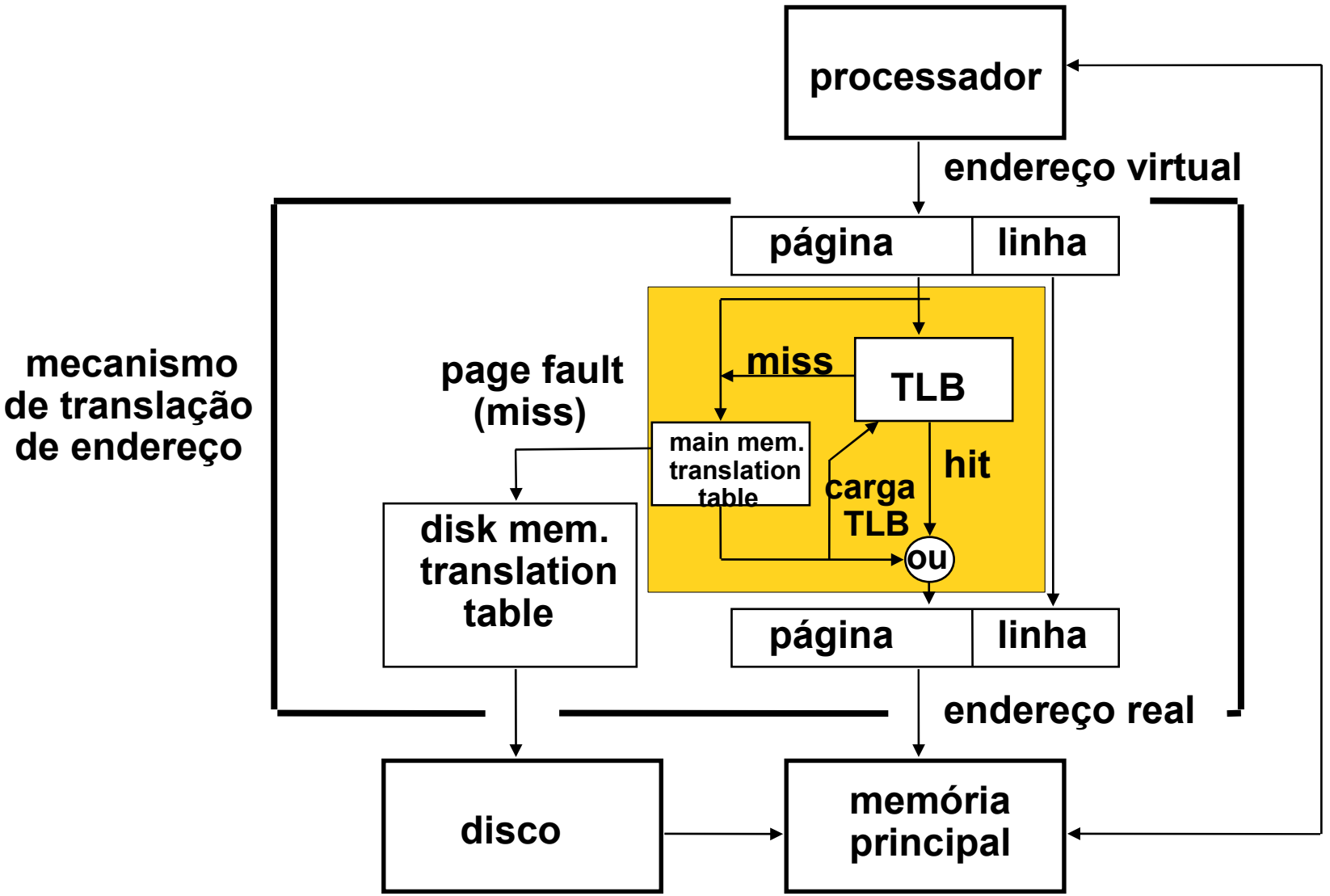
direto, associativo e conjunto - associativo

Hierarquia de Memória

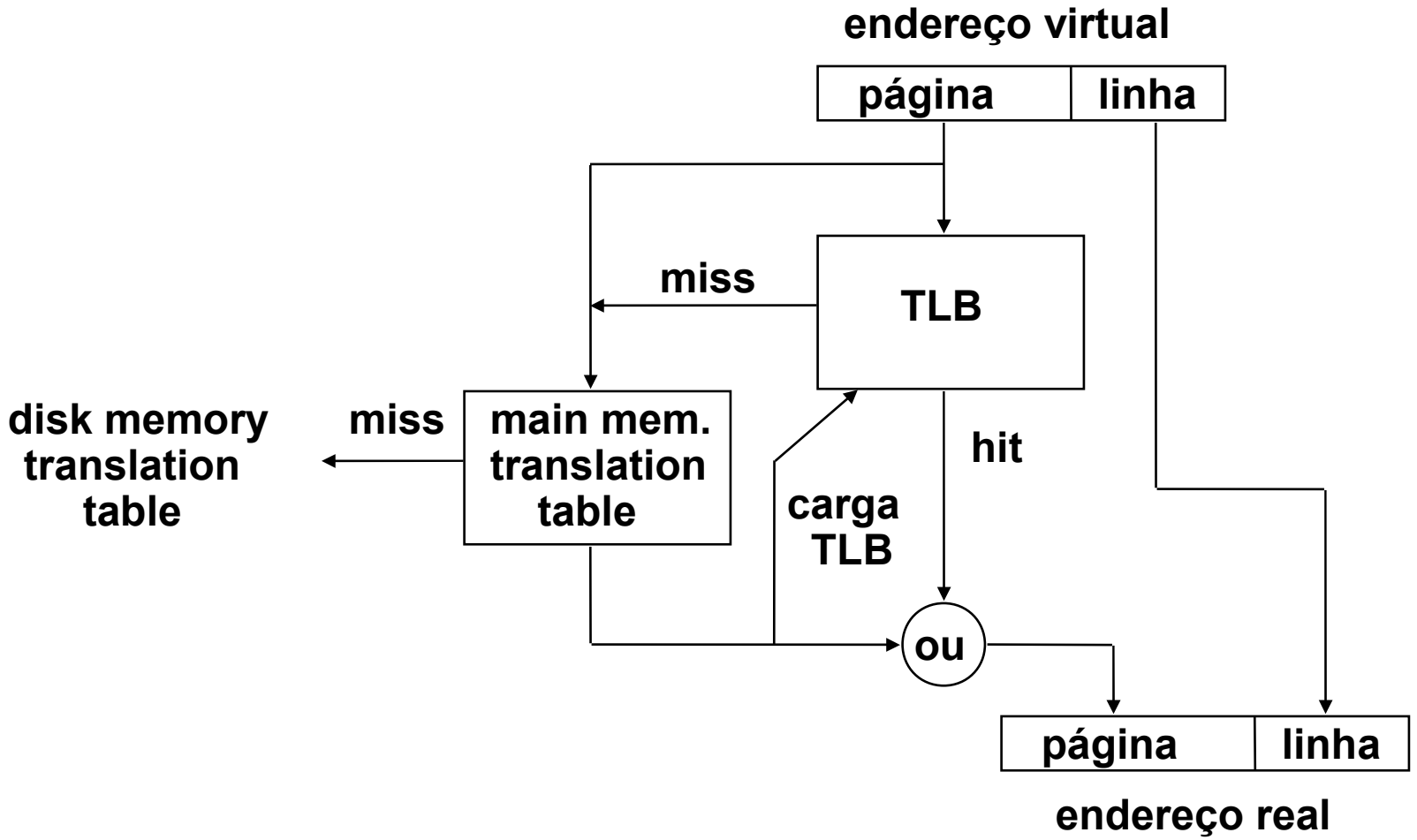


SRAM	5-25 ns
DRAM	60-120 ns
Magnetic disk	10-20 million ns

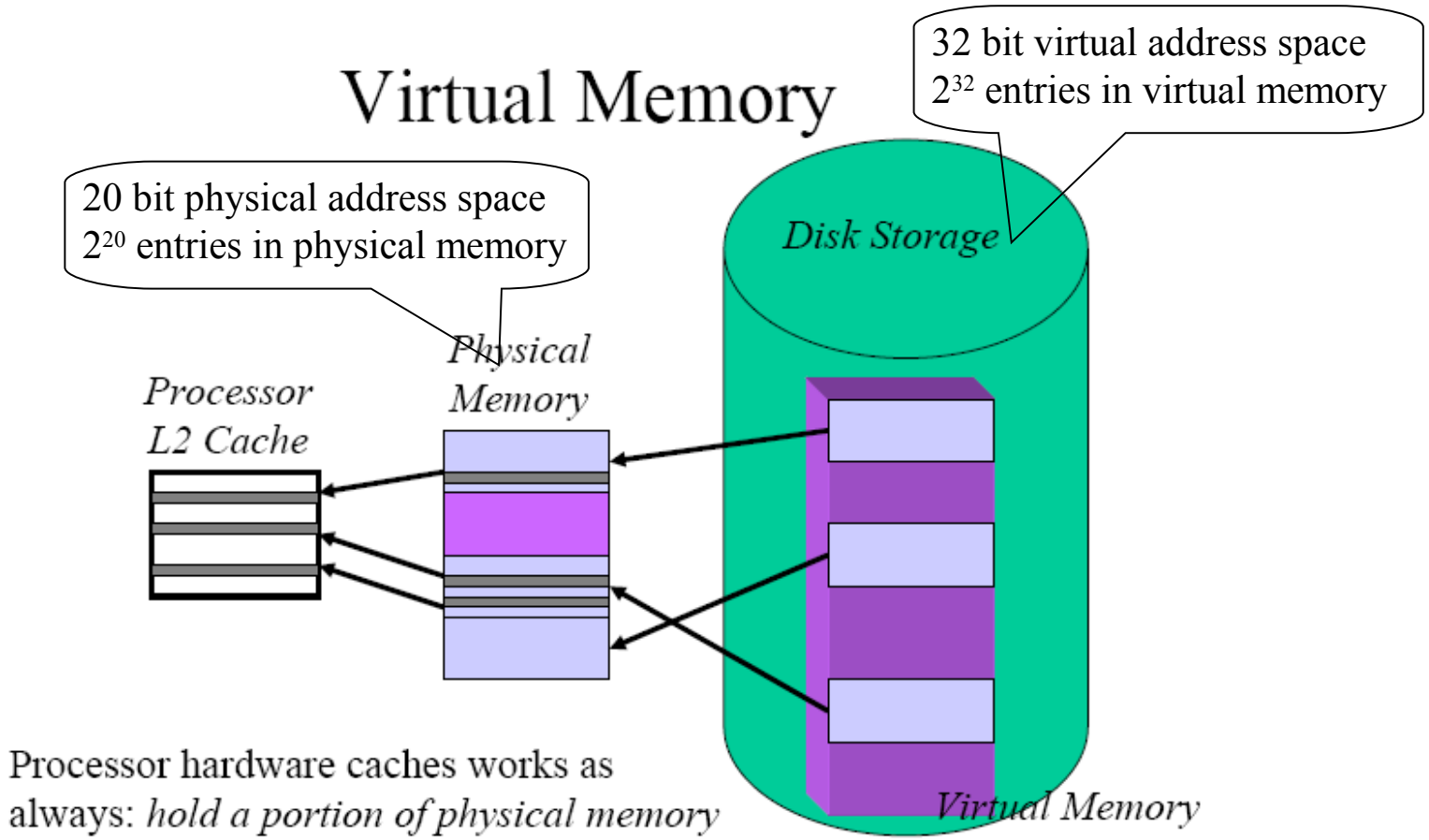
Paginação


processador
endereço virtual
página
linha
mecanismo
de translação
de endereço
page fault
(miss)
miss
TLB
main mem.
translation
table
hit
carga
TLB
ou
disk mem.
translation
table
página
linha
endereço real
disco
memória
principal

Translation Look-Aside Buffer
detalhe da página anterior

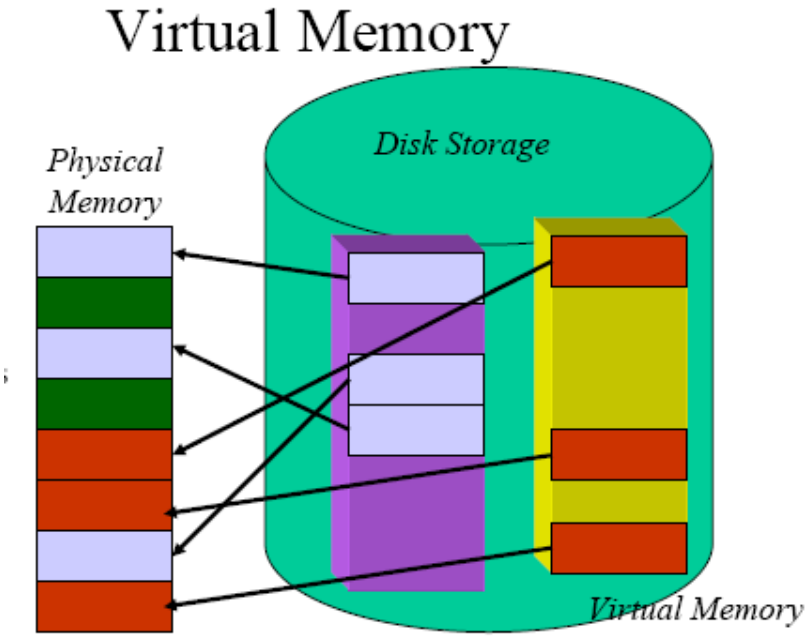


Memória Virtual



Mapeamento de dois Programas

- **Mais de um programa**
  - **Programas possuem sua própria "virtual address space"**
- **A "Virtual address space" de cada programa é mapeado para posições de memória física em tempo de execução.**
- **Não é necessário saber qual programa executará ao mesmo tempo quando o programa for compilado.**



Relocação

- **Relocação**
  - **A VM, memória virtual, é uma maneira fácil de suportar relocação,**

- **Relocação: Um programa pode ser carregado na memória física em diferentes endereços iniciais a cada vez que o programa for executado – *não contíguos***

- **Importante para multiprogramação – em tempo de execução**

- **Você não sabe o que será executado, portanto é difícil determinar o endereço de início quando o programa está sendo compilado.**

Memória virtual
segunda parte

1. Algoritmos de Substituição

2. Escrita

3. Memória Virtual + Cache

4. Segmentação

1. Algoritmos de Substituição

Page Fault

- **Page Faults**
  - **O que acontece quando a página não está na memória?**
  - ***A página deve ser encaminhada para a memória!***

- **Page fault: A página necessária não está na memória mas no disco, isto é, causando um miss**

- **Page faults são custosas:**
  - **Milhões de ciclos são necessários para um acesso ao disco**

Conseqüências do Page Fault

Soluções:

- **Grandes tamanhos de páginas:**
  - **Amortizam altos custos**
  - **Tipicamente 4KB - 64KB**
- **Redução da taxa de page fault:**
  - **Estrutura associativa**
- **Page faults tratado pelo software (SO)**
  - **uso de algoritmos inteligentes reduz a taxa de page fault**
- **Uso da estratégia write-back**
  - **Já que a estratégia write-through é cara demais**

Qual página substituir?

- **O que fazer quando ocorre Page Faults**

  Quando a página virtual não está mapeada na página física, então o SO assume o controle e carrega a página para a memória.

- **Duas ações:**

  (1) Onde está a página virtual no disco?

  (2) Onde colocar a página na memória?

- **Qual página substituir?**

  - Quando a página é buscada do disco onde colocá-la na memória principal?

- **Pegar a próxima página física não usada**

  – o SO encarrega-se da gerência das páginas livres na memória

- **Se não existe página física livre**

  – Elimina-se uma página e substitui pela página sendo carregada

1. Algoritmos de Substituição

- ***Page fault*** **ocorre se endereço da página não é encontrado nem no TLB nem na MMTT**
- **Página da memória principal é selecionada para ser substituída**
- **Substituição é demorada, pois exige acesso a disco**
  - **outro processo deve ser executado enquanto transferência é feita**
- **Página a ser substituída não estará entre aquelas registradas no TLB, pois este contém endereços de páginas utilizadas mais recentemente**
  - **TLB também deve ser atualizado após a substituição**
- **Algoritmo de substituição é executado em software (é parte do SO)**
  - **permite uso de algoritmos mais sofisticados, que reduzem alta penalidade do *page fault***

Algoritmos de Substituição

- **Substituição randômica**
  - **mais simples**
  - **usado no VAX 11/780 e no Intel i860**

- **FIFO (first-in first-out)**
  - **fila de páginas pode ser mantida em software**
  - **adaptação: passar por cima de páginas que tenham sido referenciadas recentemente**

- **LRU (least recently used)**
  - **aproximação: manter um bit *use* para cada página**
    - **a intervalos fixados pelo S.O., bits de todas as páginas são testados e resetados**
    - **registro do nº de vezes em que bit foi encontrado igual a 1 dá idéia aproximada da utilização**

2. Escrita

2. Escrita

- ***Write-through*** **não pode ser utilizado devido ao alto custo do acesso ao disco**

- **Copiar uma página completa é bem mais eficiente do que copiar blocos pequenos (palavras ou linhas)**

- **Uso de 1 bit *modified* associado a cada página de uma *page table* ou TLB**
  - **bit é setado quando é feita escrita na página**
  - **página não precisa ser escrita de volta no disco ao ser substituída, caso não tenha sido modificada desde o fetch**

3. Memória Virtual + Cache

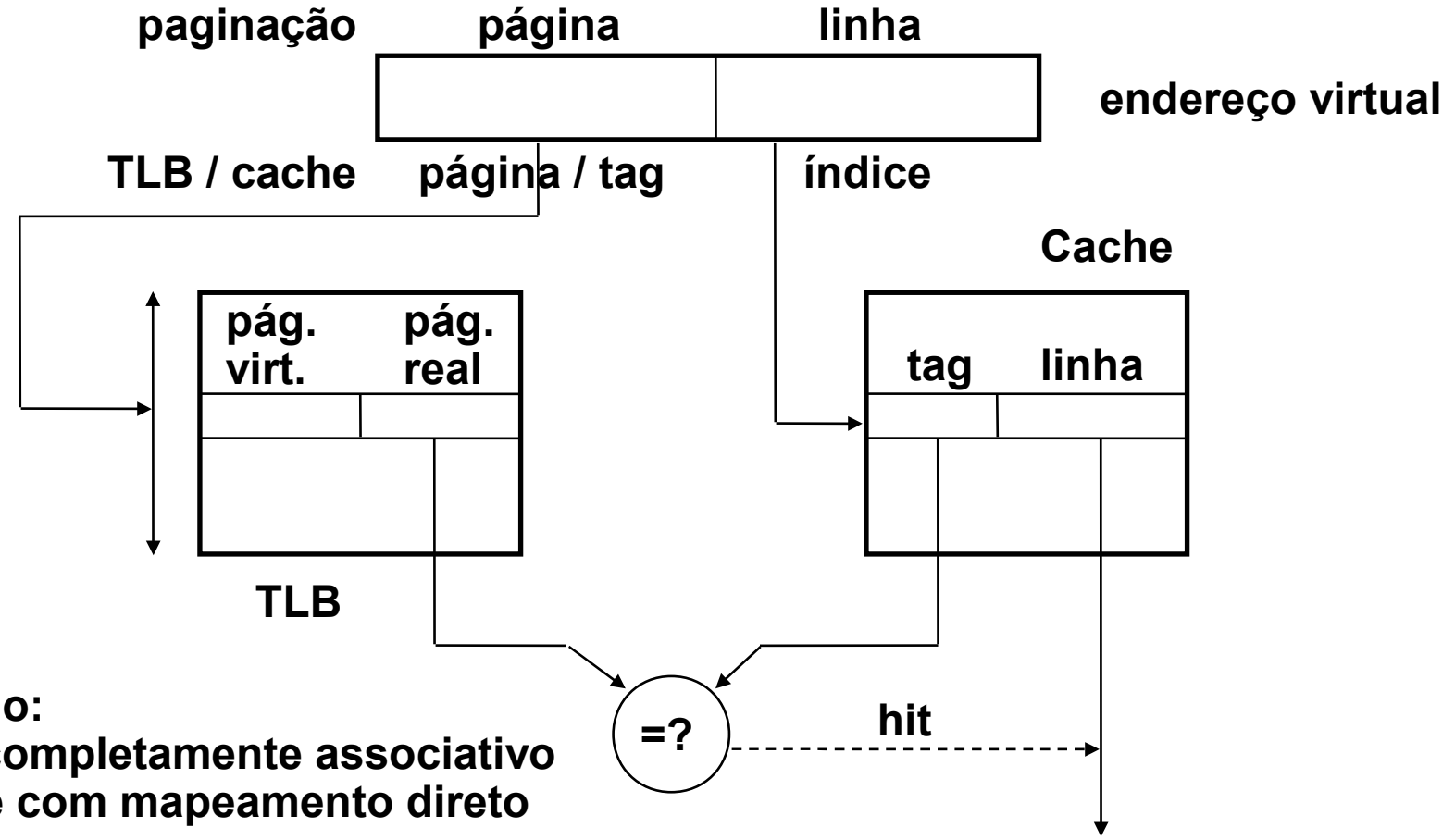
3. Memória Virtual + Cache

- **Integrando o Sistema de Memória Virtual com as Caches**

- **Se sistema tem memória virtual e cache, cache pode ser colocada ...**
  - **após a translação de endereço virtual em endereço real pelo TLB**
  - **antes da translação de endereço**

- **Colocando cache após translação**
  - **cache contém tags de endereços reais**
  - **superposição entre translação e acesso à cache**
  - **tamanho da página = tamanho da cache, supondo cache com mapeamento direto**
  - **solução mais simples**

Cache após Translação



supondo:
– TLB completamente associativo
– cache com mapeamento direto

Tempo de Acesso

- **Se há miss no TLB, translação é feita através de *page table* que pode estar na cache**

- **6 combinações possíveis, supondo cache endereçada com endereço real (cache após TLB)**
  - **página está no TLB, linha está na cache**
  - **página está no TLB, linha está na memória principal**
  - ***page table* está na cache, linha está na cache**
  - ***page table* está na cache, linha está na memória principal**
  - ***page table* está na memória principal, linha está na cache**
  - ***page table* está na memória principal, linha está na memória principal**

Cache e Memória Virtual

- **Alpha 21164**
  - **cache de instruções endereçada virtualmente**
  - **cache de dados endereçada fisicamente**
- **Pentium II**
  - **caches de instruções e dados endereçadas fisicamente**
- **HP PA-2**
  - **cache única endereçada virtualmente**

4. Segmentação

4. Segmentação

- **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 contíguos e tamanhos variáveis**

- **Endereço dividido em 2 partes**
  - **número do segmento (ou base)**
  - **deslocamento (ou *offset*)**

- **Segmento e deslocamento devem ser somados, e não concatenados**

- **Endereço virtual = endereço lógico**

- **Endereço real = endereço físico**

Paged and Segmented VM

- **Virtual memories can be catagorized into two main classes**
  - **Paged memory : fixed size blocks**
  - **Segmented memory : variable size blocks**

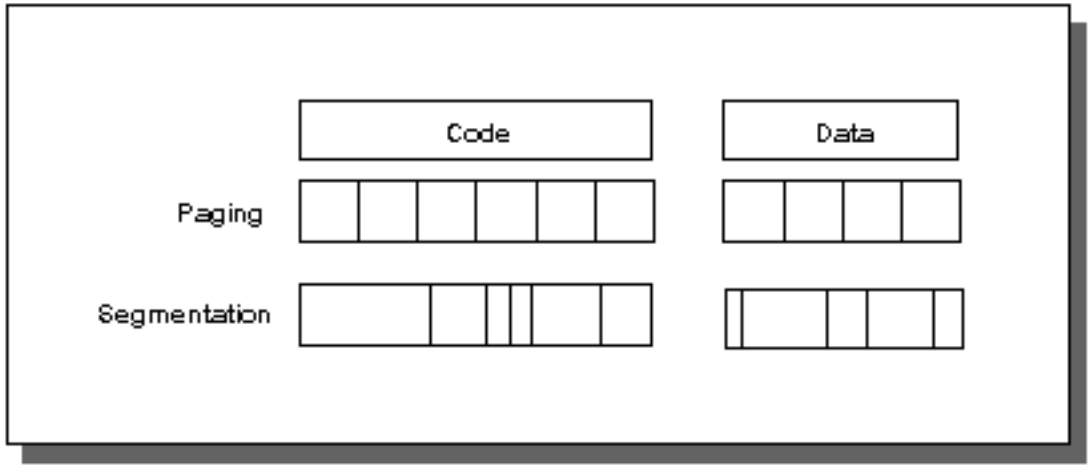


FIGURE 5.38 Example of how paging and segmentation divide a program.

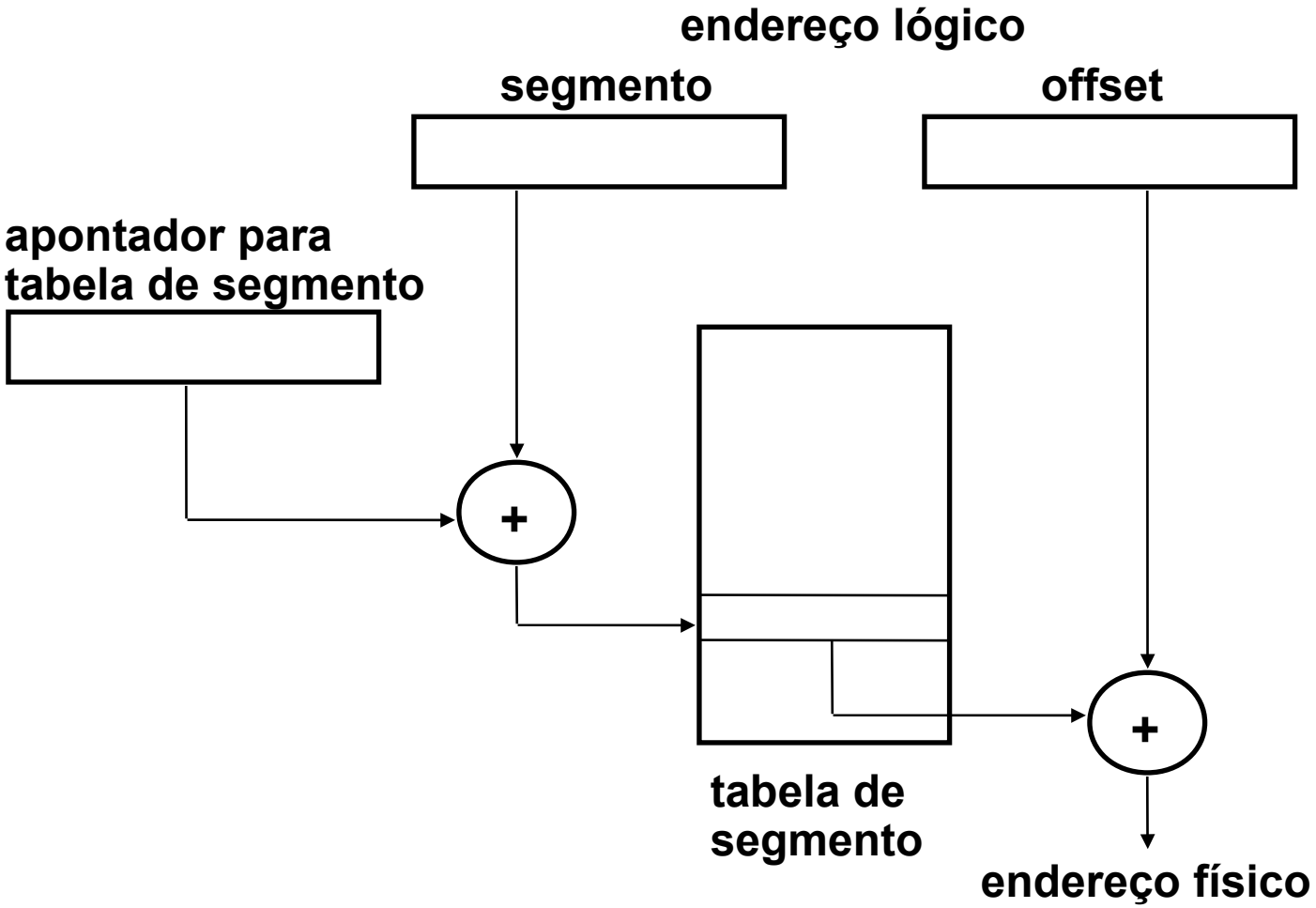
Paged vs. Segmented VM

- **Paged memory**
  - **Fixed sized blocks (4 KB to 64 KB)**
  - **One word per address (page number + page offset)**
  - **Easy to replace pages (all same size)**
  - **Internal fragmentation (not all of page is used)**
  - **Efficient disk traffic (optimize for page size)**

- **Segmented memory**
  - **Variable sized blocks (up to 64 KB or 4GB)**
  - **Two words per address (segment + offset)**
  - **Difficult to replace segments (find where segment fits)**
  - **External fragmentation (unused portions of memory)**
  - **Inefficient disk traffic (may have small or large transfers)**

- **Hybrid approaches**
  - **Paged segments: segments are a multiple of a page size**
  - **Multiple page sizes: (e.g., 8 KB, 64 KB, 512 KB, 4096 KB)**

Translação de Endereços de Segmentos



Segmentação

- **Translação utiliza normalmente mapeamento direto**
- **Uma tabela de segmento para cada processo ativo**
- **Registrador especial contém endereço inicial da tabela de segmento**
- **Tabela de segmento contém**
  - **comprimento do segmento**
  - **bits de proteção de memória**
  - **bits para o algoritmo de substituição**
- **Comprimento do segmento**
  - **armazenado na tabela de segmento para evitar que programa acesse erradamente posições fora do segmento**

Segmentação

- **Proteção de memória: segmento pode ser**
  - **read-only**
  - **execute-only**
  - **system-only**

- **Algoritmo de substituição**
  - **mais complexo do que em paginação devido ao tamanho variável dos segmentos**
  - **problemas de fragmentação de memória**

- **Combinação de paginação e segmentação**
  - **Pentium**

FIM