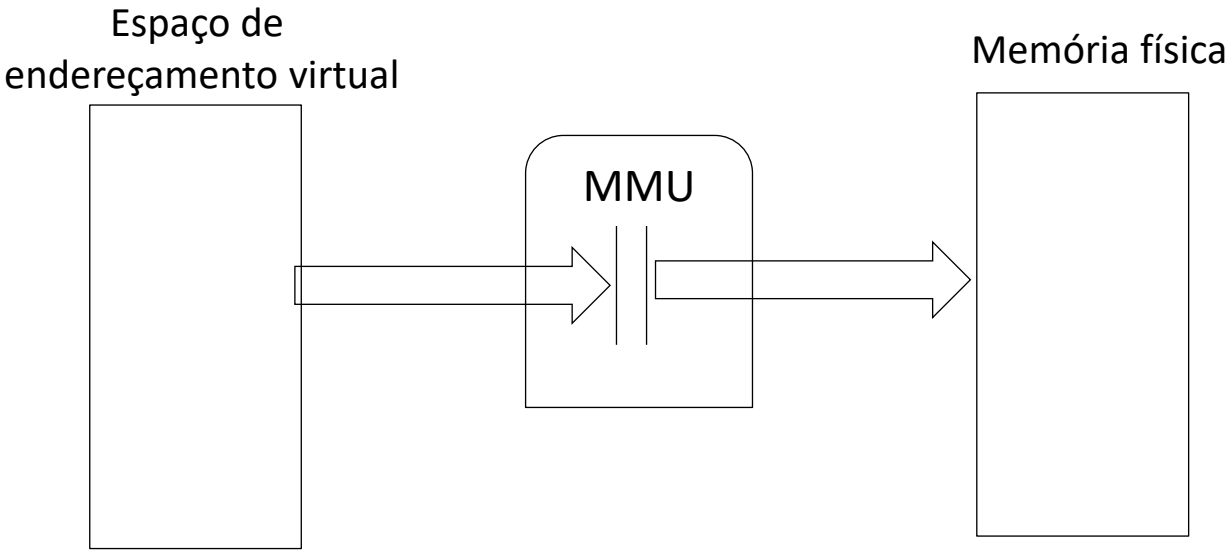
Gestão de memória

- Os sistemas operativos modernos de utilização genérica oferecem uma abstração do recurso memória que permite aos processos:
  - Ter a ilusão de que a memória é de seu uso exclusivo, isto é, os processos só podem aceder aos seus dados/código, não podendo interferir com os dados/código de outros processos (exceto através da partilha combinada de memória).
  - Ter a ilusão de que a memória disponível pode ser superior à RAM existente na máquina, abstração designada como memória virtual (através da cópia, de forma transparente aos processos, de zonas não utilizadas de momento para memória secundária, de modo a libertar espaço de RAM para outros processos ou para o próprio).
- Para suportar esta abstração é necessário suporte *hardware*, através da chamada MMU (Memory Management Unit).

Arquitecturas de MMU (Memory Management Unit)

- A abstração da memória baseia-se na distinção entre Espaço de Endereçamento Virtual (os endereços acedidos/emitidos pela execução do programa) e Espaço de Endereçamento Físico (endereços da RAM).
- É necessário suporte *hardware* para a transformação necessária entre endereços virtuais e endereços físicos, a chamada MMU.
  - Arquitectura Segmentada (1ª solução)
  - Arquitectura Paginada (solução atual)

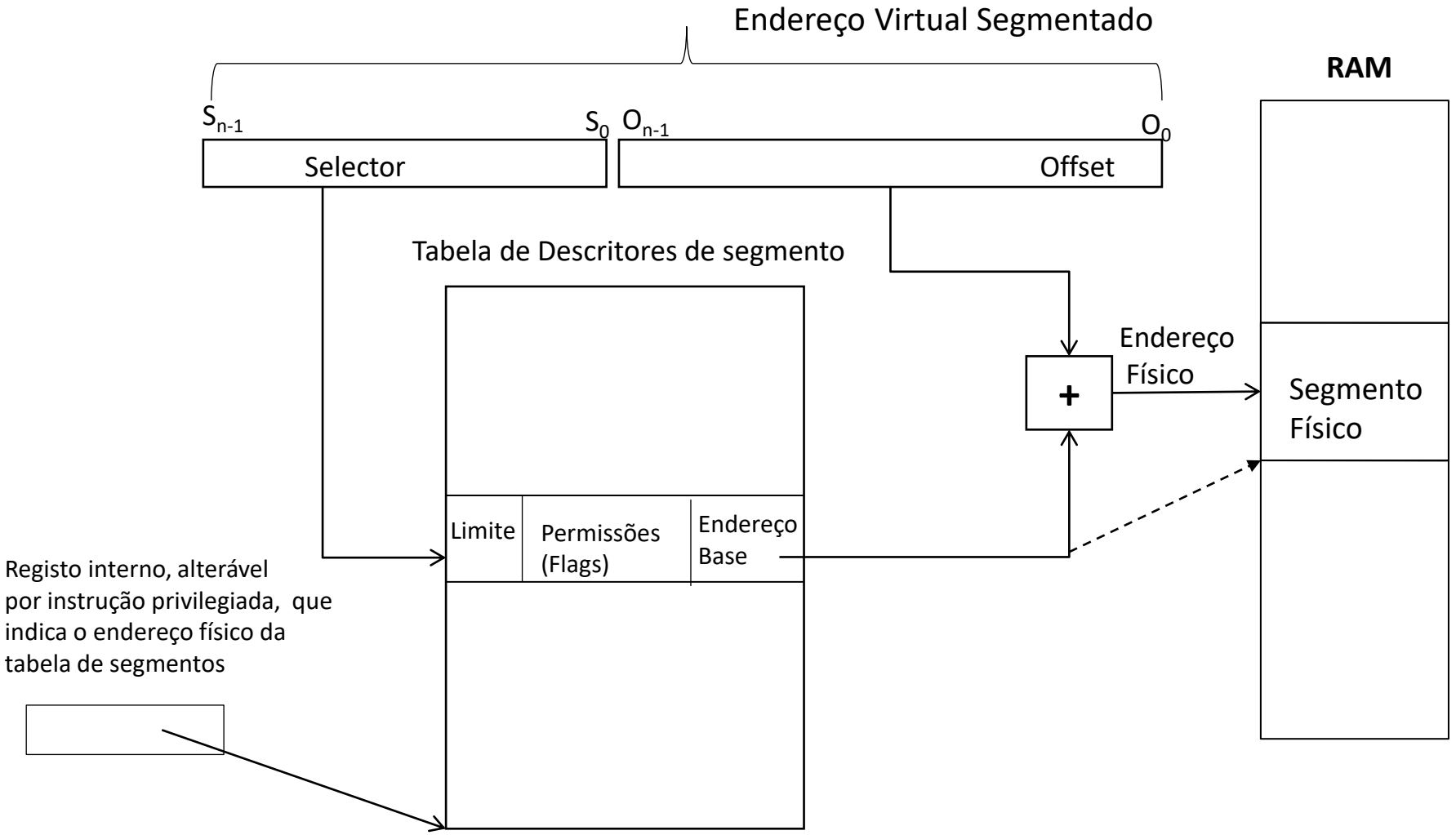


Arquitectura segmentada

- Na arquitectura segmentada o espaço de endereçamento virtual é constituído por um número arbitrário de segmentos de código, dados e *stack*, com dimensões variáveis.
  - Os segmentos são visíveis às aplicações e resultam de um modelo de programação orientada à organização do código e dados em módulos, ocupando segmentos distintos.
  - Um endereço virtual é especificado por duas componentes: O **`identificador`** do segmento (normalmente chamado de seletor), correspondente ao índice de uma tabela de descritores de segmento, e o **`offset`** dentro do segmento.

Segmentos aplicacionais dentro de um processo.O espaço virtual é segmentado

Arquitectura Segmentada (Segmentação)

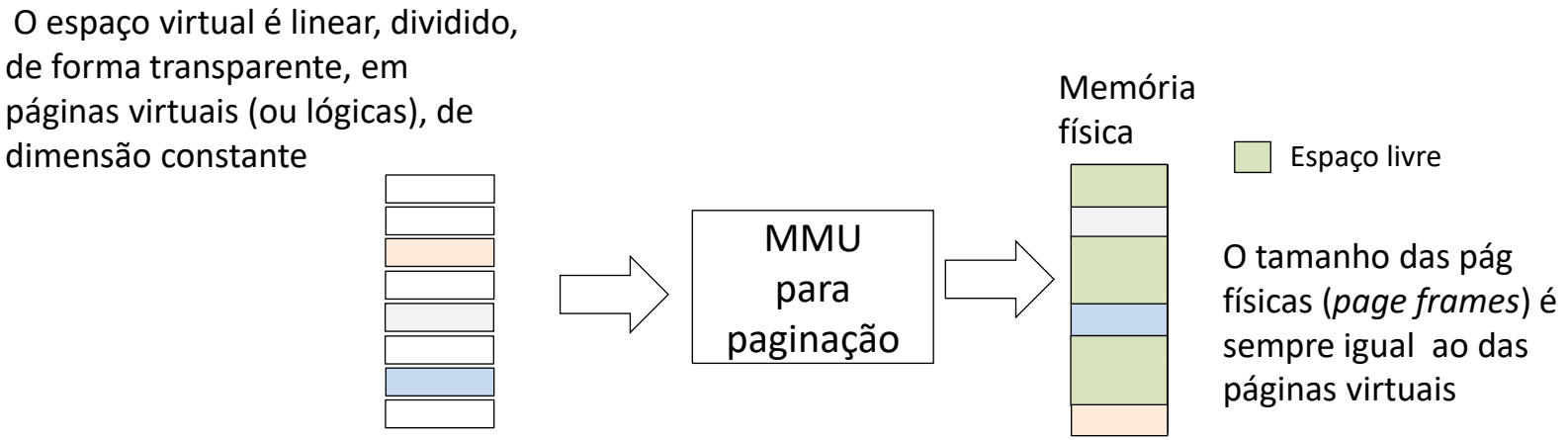


Características da Segmentação

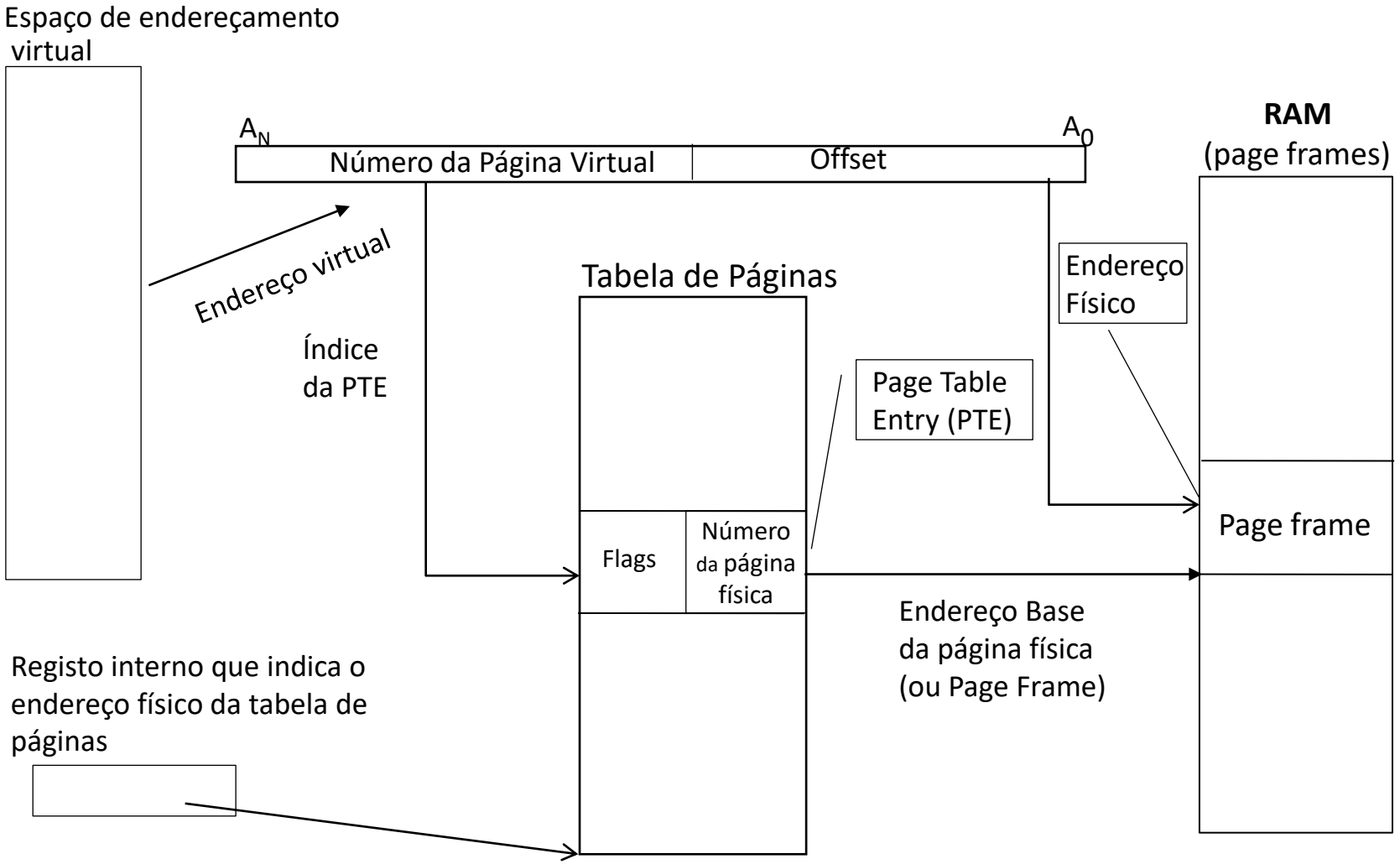
- **Vantagens**
  - Mapeamento simples o que reduz a complexidade do *hardware*. A associação de uma cache de segmentos em uso evita o acesso constante à tabela de descritores.
  - Facilita a recolocação de código e dados.
  - Facilita a criação de regiões de memória partilhada entre processos.
- **Desvantagens**
  - Modelo de ponteiros complexo.
  - Gestão complexa da memória disponível. O facto dos blocos a gerir terem dimensões variáveis gera problemas já vistos na implementação de *heaps*, nomeadamente a fragmentação externa.
  - Implementação de memória virtual pouco eficiente, devido ao tempo dispendido em escritas/leituras do disco para segmentos grandes (> 64KB)

Arquitectura paginada

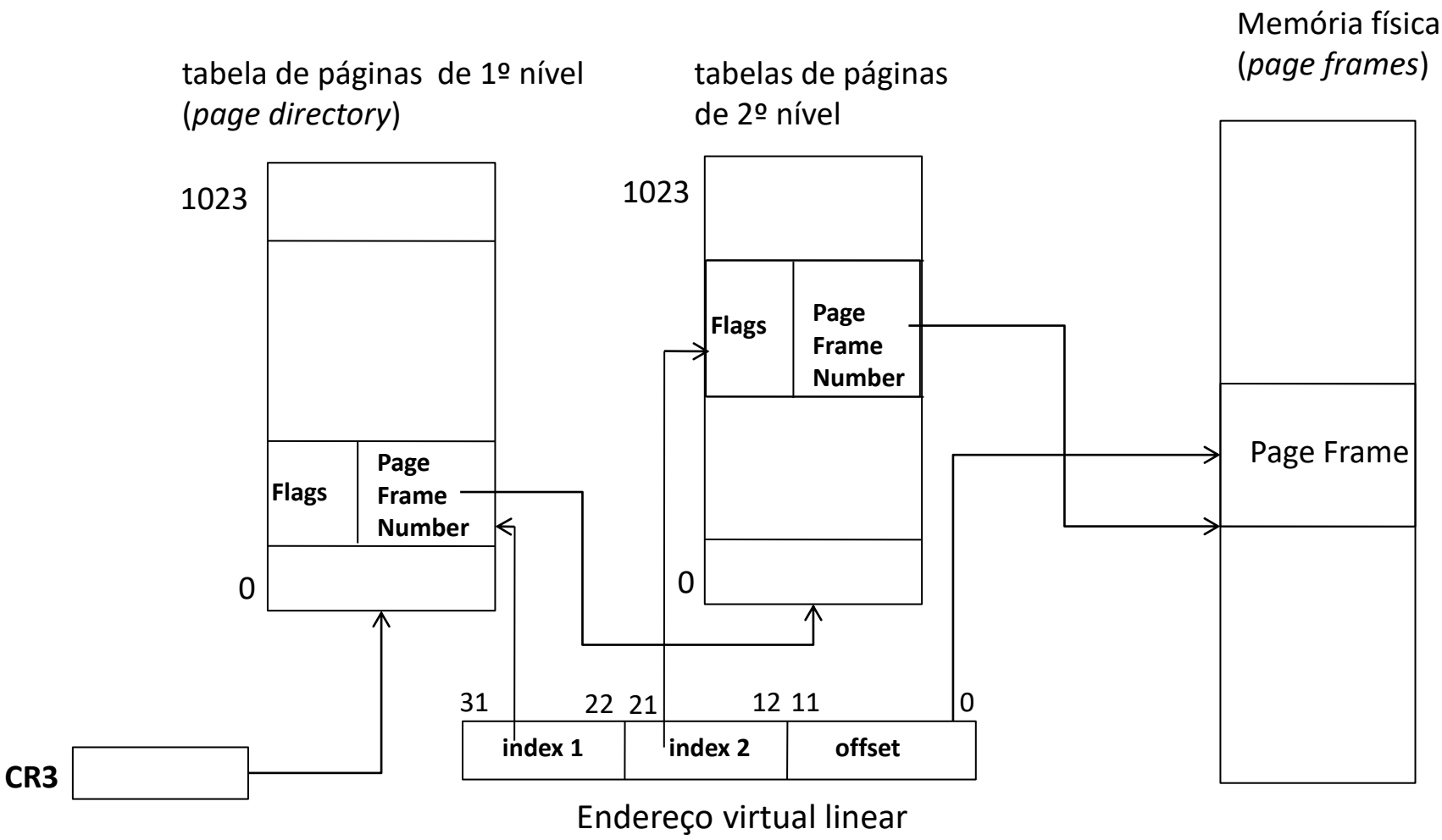
- A MMU paginada organiza o espaço de endereçamento virtual e físico em blocos de dimensão fixa (páginas virtuais e físicas).
  - Em diferentes MMU a dimensão das págs. pode ser diferente (4KB, 8KB, etc.), mas as páginas virtuais e páginas físicas de uma dada MMU têm sempre a mesma dimensão.
- Na arquitectura paginada as páginas são transparentes às aplicações:
  - O espaço de endereçamento que as aplicações vêem é linear
  - A organização lógica nada tem a ver com a organização física (ex: uma instrução ou uma variável global podem começar no final de uma página e terminar na página seguinte ocupando endereços físicos não adjacentes.



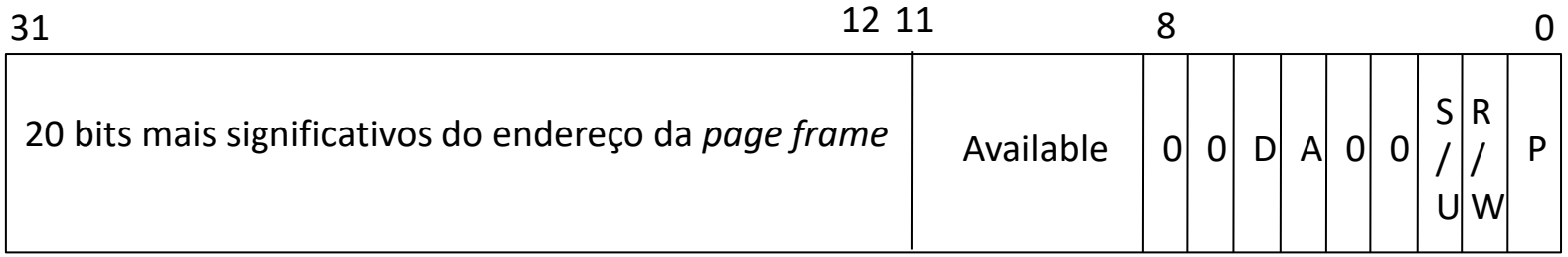
Arquitectura Paginada (Paginação)



Arquitectura Paginada
(Necessidade de arquitecturas multinível - exemplo para x86-32)



Entrada de tabela de páginas – PTE (Page Table Entry)
(exemplo para i386)



P – Present (indica que a página está associada a uma page frame)
R/W – Read/Write (indica se a página pode ser modificada)
S/U – Supervisor/User (indique que a página pode ser usada em user mode)
0 – bits reservados (já utilizados em versões mais recentes do CPU)

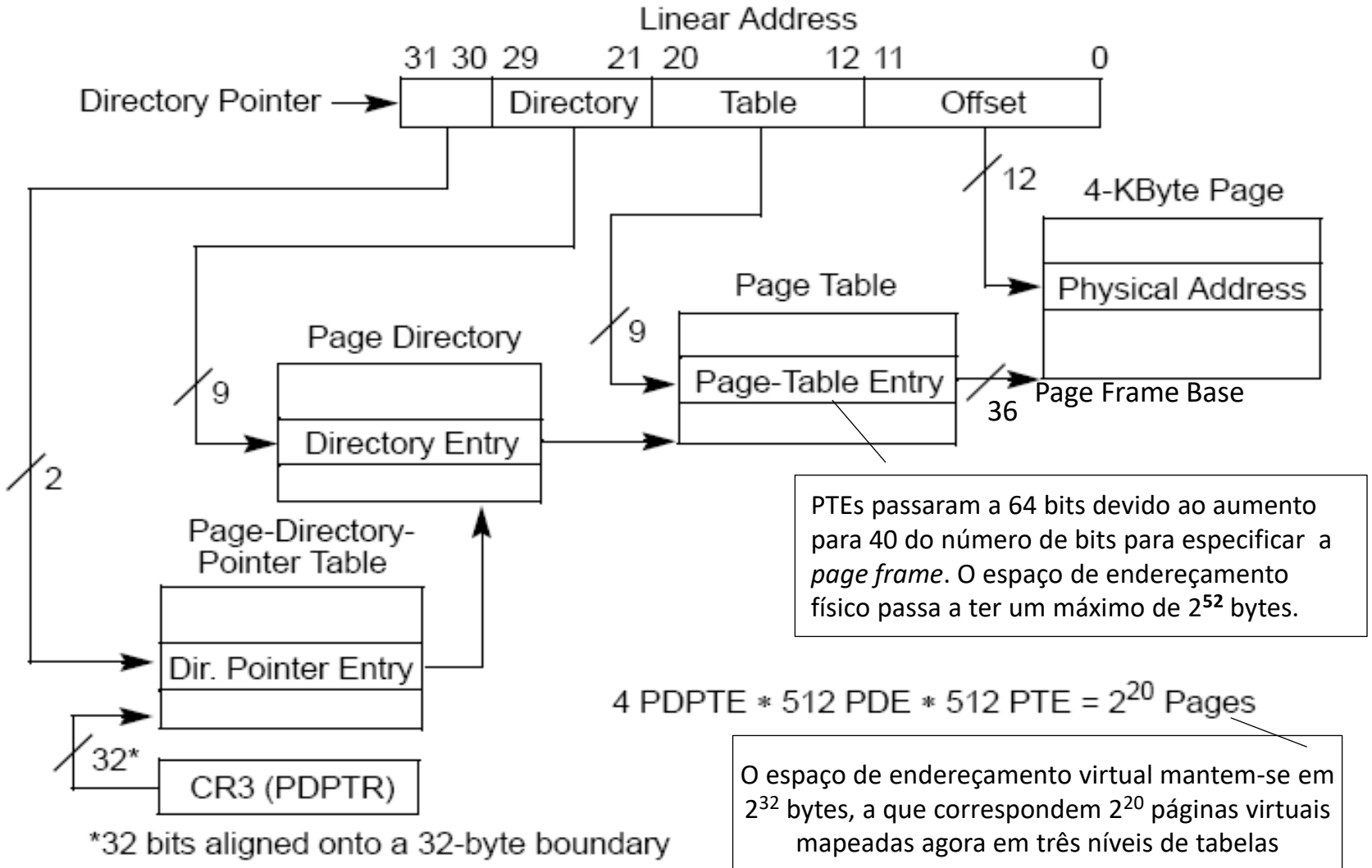
Bits alterados automaticamente pela MMU:
A – Acessed (indica que a página for acedida)
D – Dirty (indica que a página foi modificada

Available – Disponíveis para uso pelo Sistema Operativo

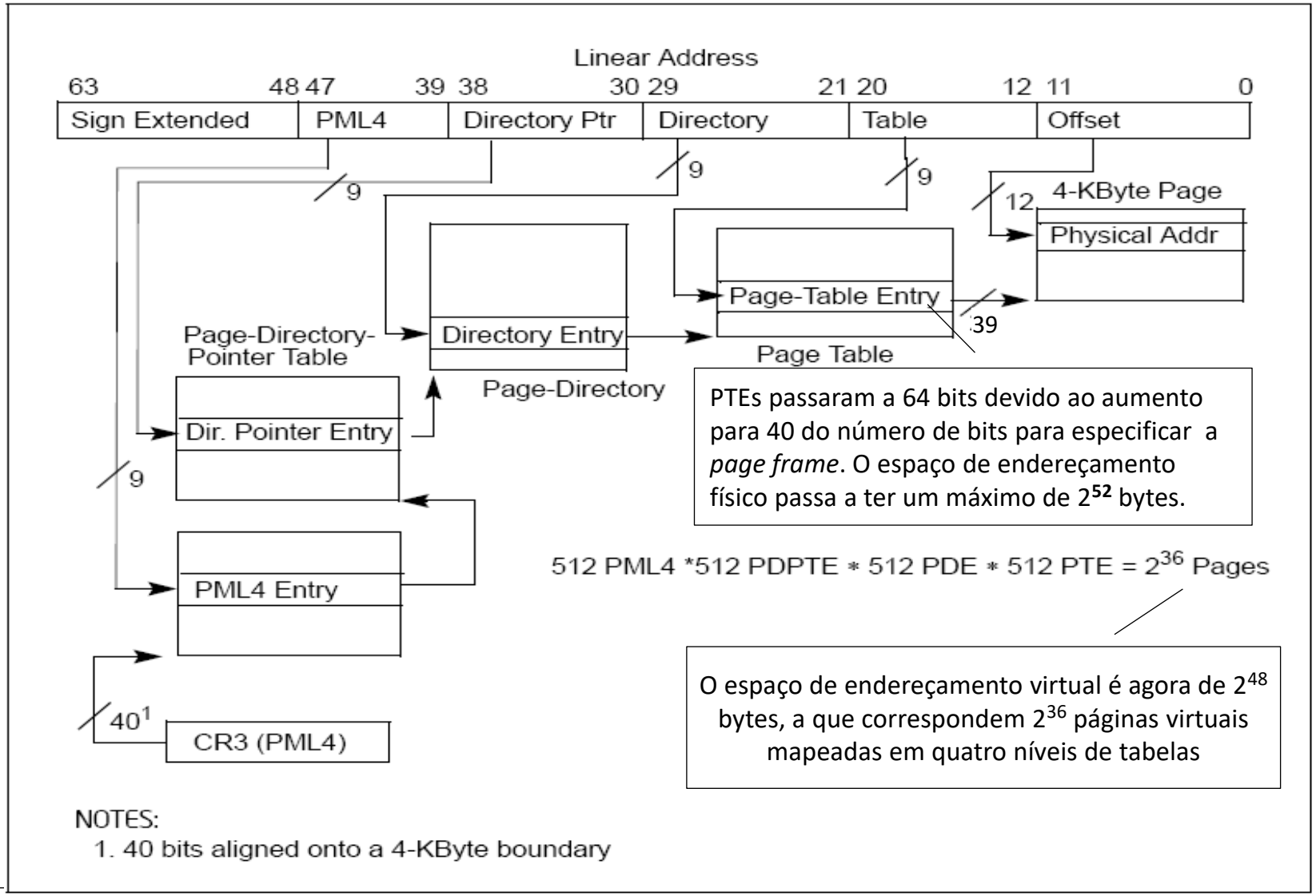
Exercício

- Considere um processador com endereços virtuais de 34 bits que utiliza uma estrutura de paginação de dois níveis. Os endereços virtuais são divididos em: 10 bits para a tabela de 1º nível e 11 bits para as tabelas de 2º nível.
  Sabendo que as tabelas de páginas ocupam sempre uma página, indique:
  - Qual a dimensão das páginas e de cada PTE?
  - Quantas páginas existem no espaço de endereçamento virtual?
  - Sabendo que existem 10 bits na PTE para especificar a page frame qual o espaço de endereçamento físico?
  - Admitindo a existência de 10 bits de controlo (flags) em cada PTE, qual a dimensão máxima do espaço de endereçamento físico, possível nesta arquitetura?

Variantes em CPU´s modernos da família x86 - **Physical Address Extensions (PAE)**

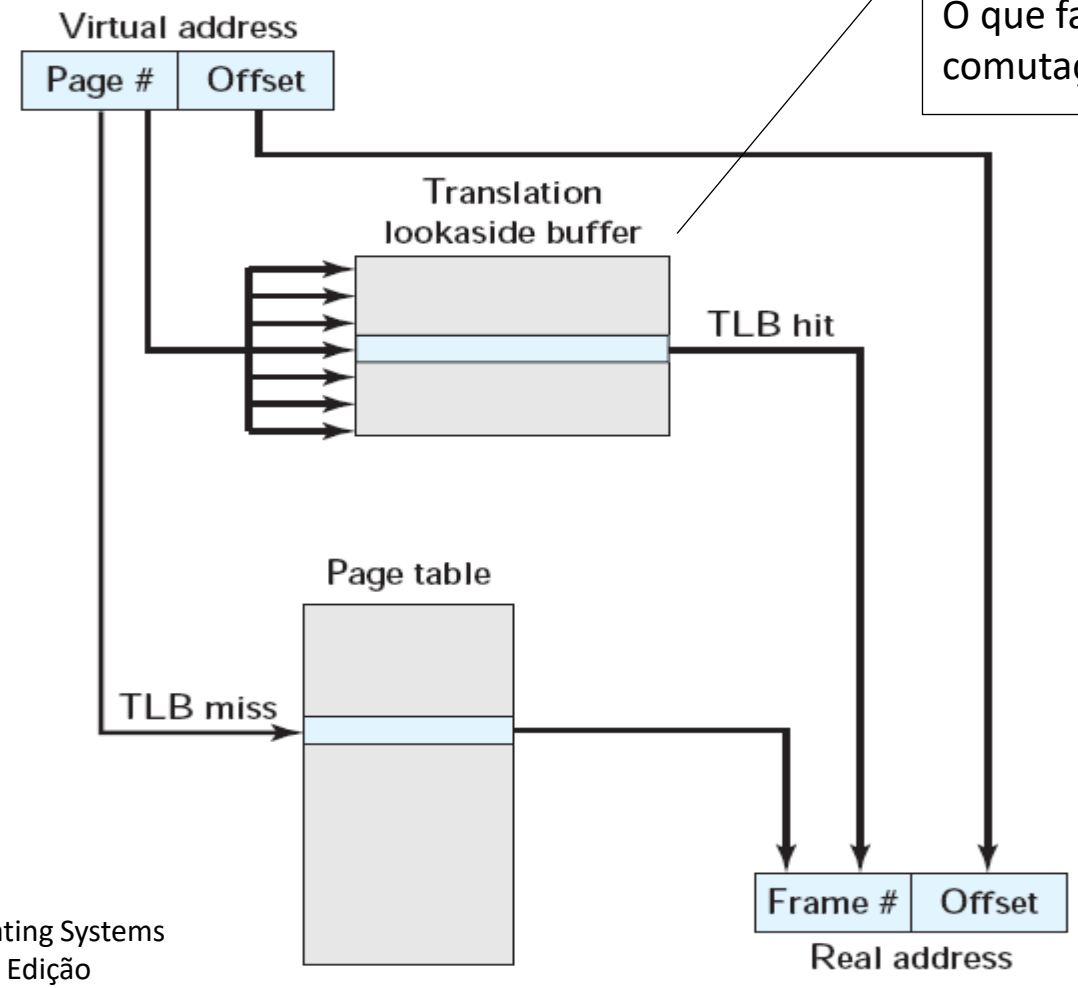


Paginação na arquitectura x64



Paginação – A necessidade da cache TLB

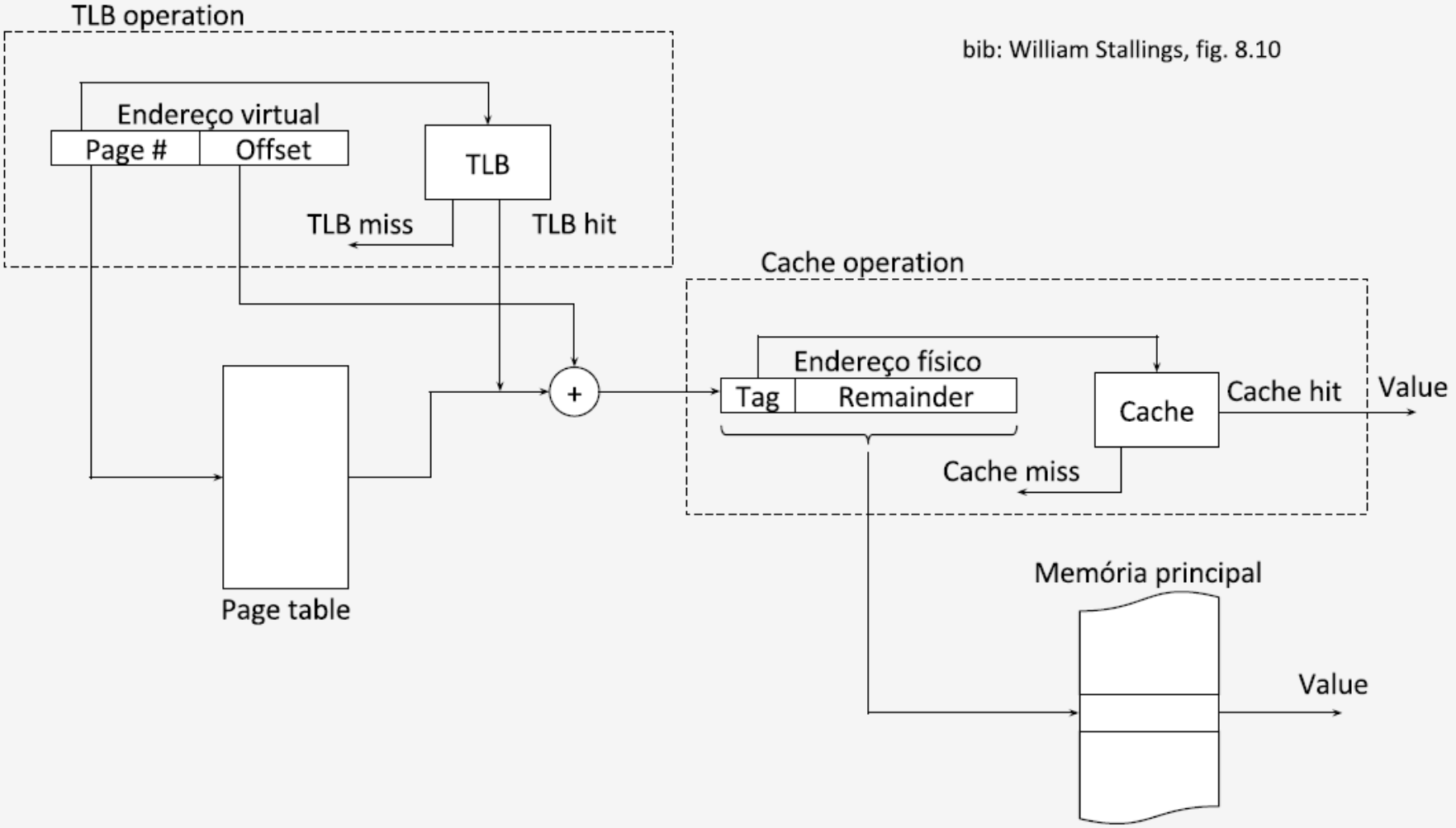
(Translation Lookaside Buffer)



O que fazer à TLB na comutação de processos?

Retirado do livro Operating Systems
de William Stallings, 6ª Edição

Cache (de PTE's e de dados) – visão global



Objectivos de aprendizagem

- Compreender e enunciar as principais características das arquitecturas segmentada e paginada.
- Identificar vantagens/desvantagens de cada um dos modelos.
- Nas arquitecturas paginadas:
  - explicar a necessidade de existência de arquitecturas multi-nível
  - explicar a necessidade da cache TLB
  - sintetizar características da arquitectura (dimensões de PTE, de página (lógica e física), de espaços de endereçamento virtual e memória física, número de níveis, etc.), a partir de informações parcelares sobre a arquitectura.
  - Enunciar e compreender as *flags* típicas presentes num PTE
  - justificar as dimensões típicas das páginas nas arquitecturas actuais.

Bibliografia

- Tanembaum, Modern Operating Systems, 4ª Ed.
  - Cap. 3, Memory Management (3.1,3.2, 3.3)