Abstrações de Memória:
Memória Virtual e Paginação

Prof. Dr. Márcio Castro
marcio.castro@ufsc.br

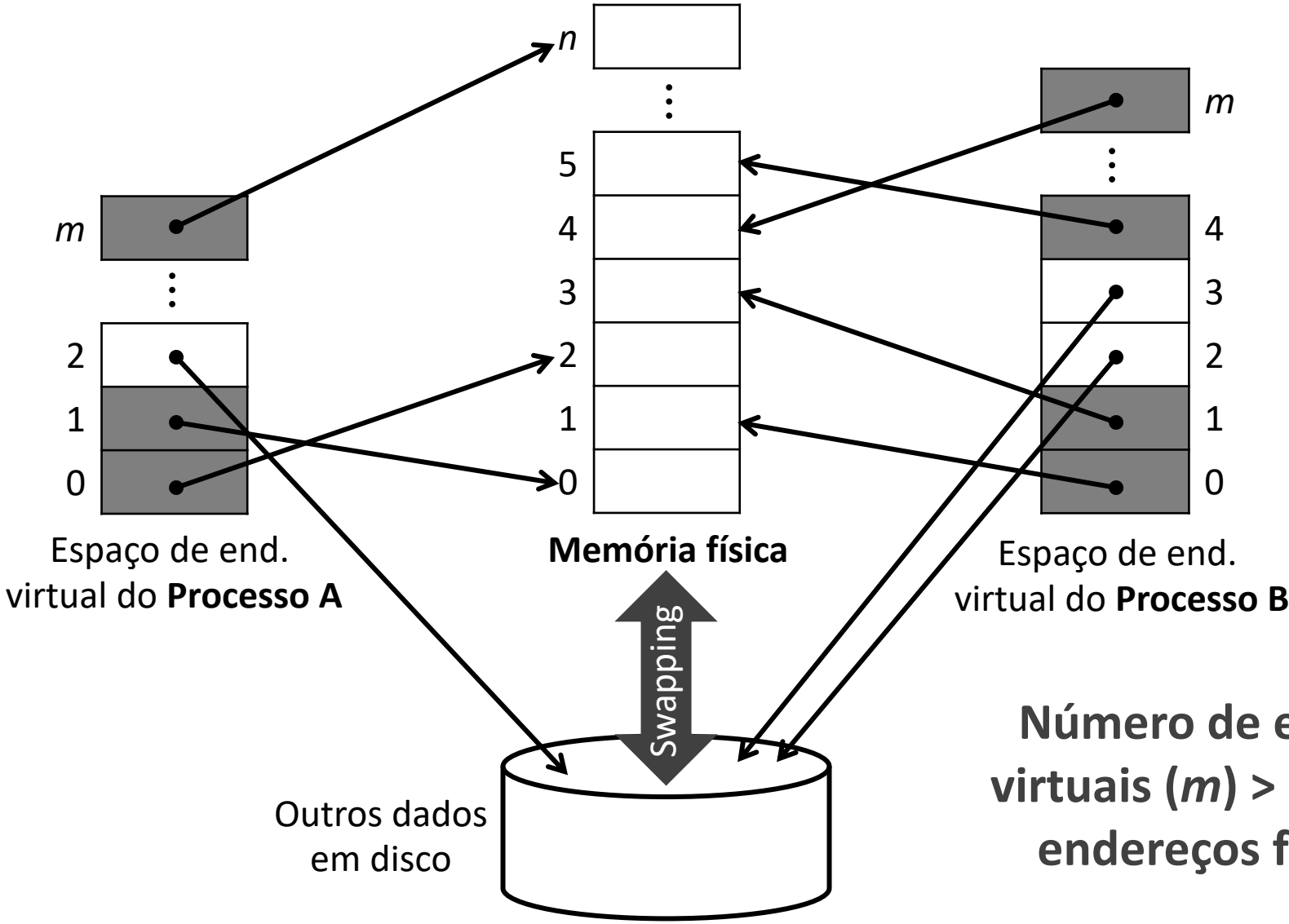
UNIVERSIDADE FEDERAL
DE SANTA CATARINA

1 Memória virtual

Memória virtual

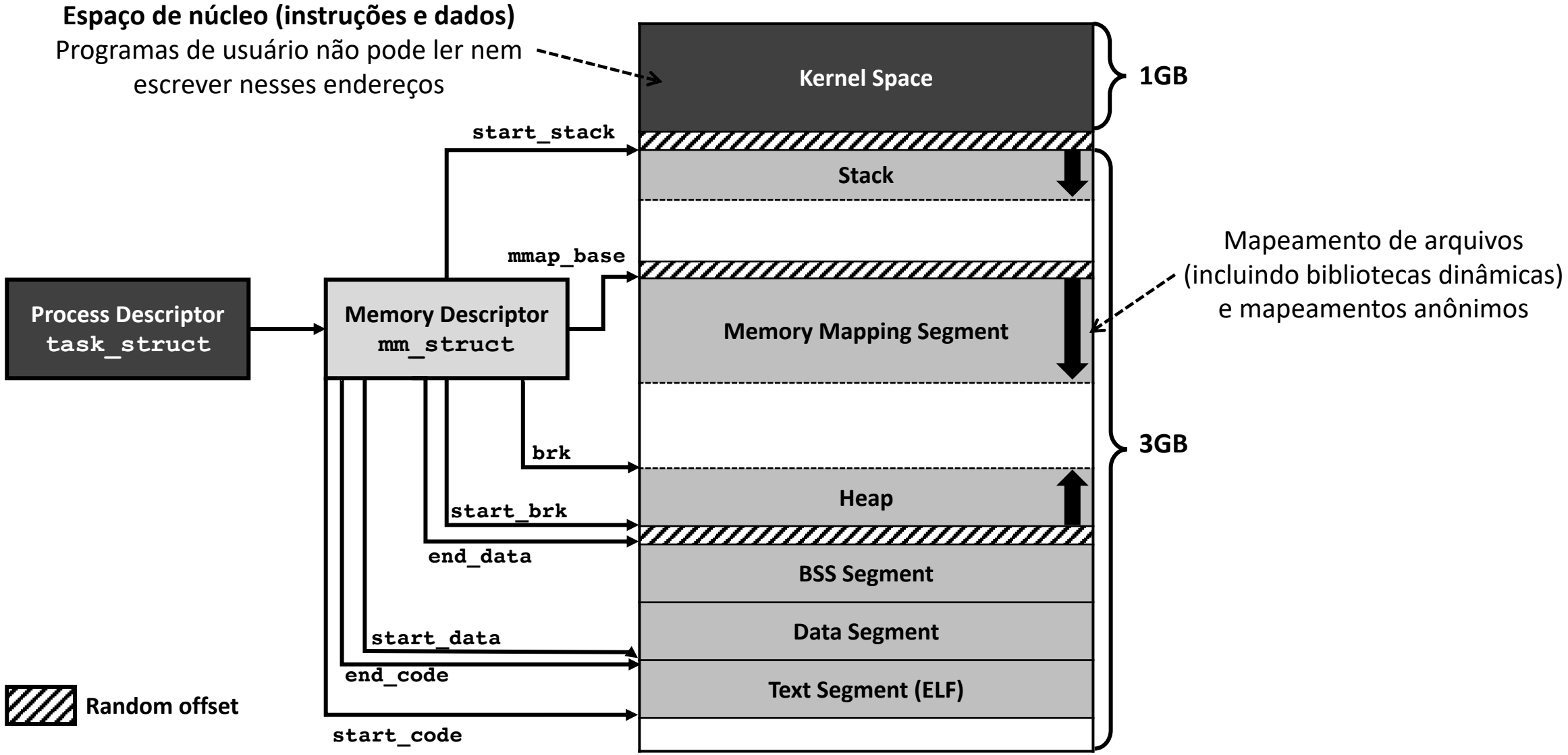
- **Virtualização de endereços físicos**
  - Endereços gerados pelos processos são **virtuais**, e constituem o **espaço de endereçamento virtual** do processo
  - Espaço de endereçamento virtual é **contíguo começando pelo endereço 0**
  - Endereços virtuais são **mapeados para endereços físicos**
  - Não havendo espaço na RAM, os dados são armazenados no disco (**swapping**)

Memória virtual



Número de endereços virtuais (*m*) > número de endereços físicos (*n*)

Espaço de endereçamento virtual no Linux (32 bits)



2 Paginação

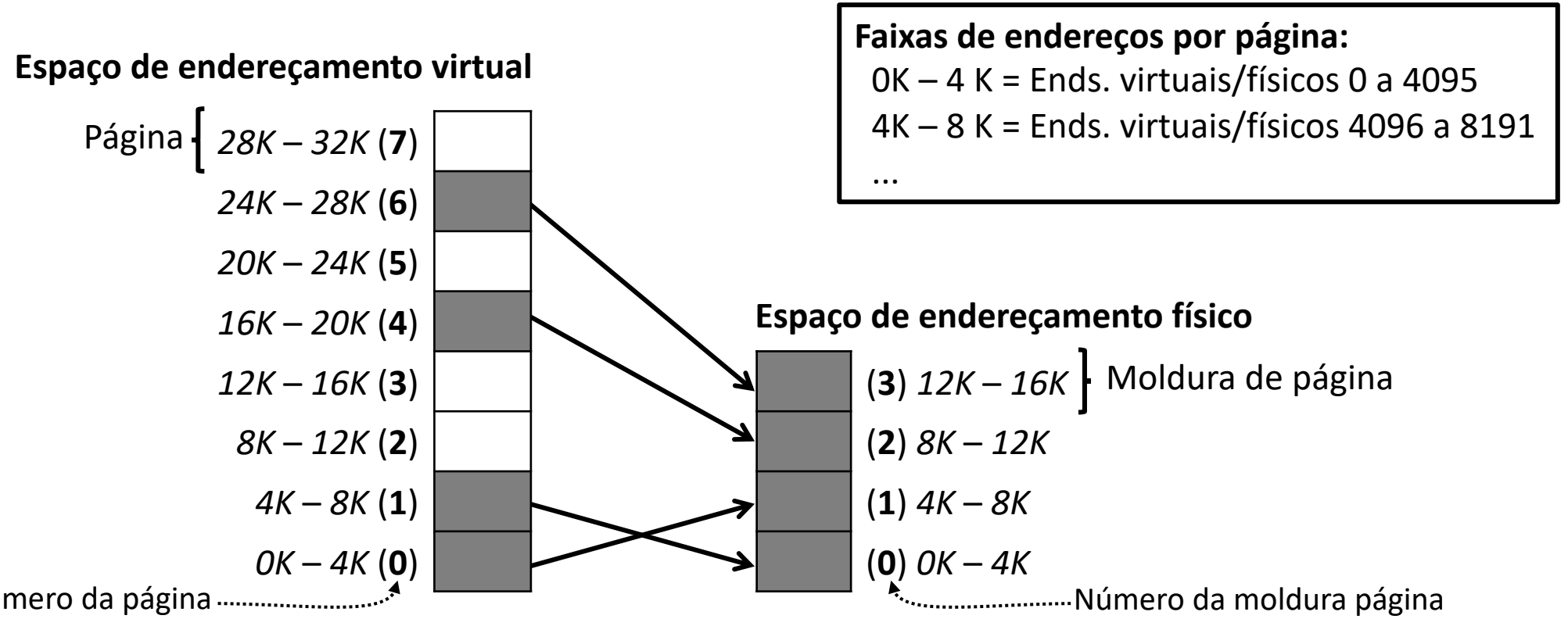
Memória virtual com paginação

- A maioria dos sistemas de memória virtual usa uma técnica denominada de **paginação**
- **Na paginação**
  - O espaço de endereçamento virtual consiste em unidades de armazenamento de **tamanho fixo** denominadas **páginas[1] (ou páginas virtuais)**
  - Unidades correspondentes no espaço de endereçamento físico (RAM) são denominadas **molduras de página[2] (ou páginas físicas)**
  - **Páginas** e **molduras de página** possuem o **mesmo tamanho**
  - O SO mantém em memória uma **tabela de páginas** para cada processo, a qual relaciona **páginas** com **molduras de página**

[1] *Em inglês: pages / virtual pages*
[2] *Em inglês: page frames / physical pages*

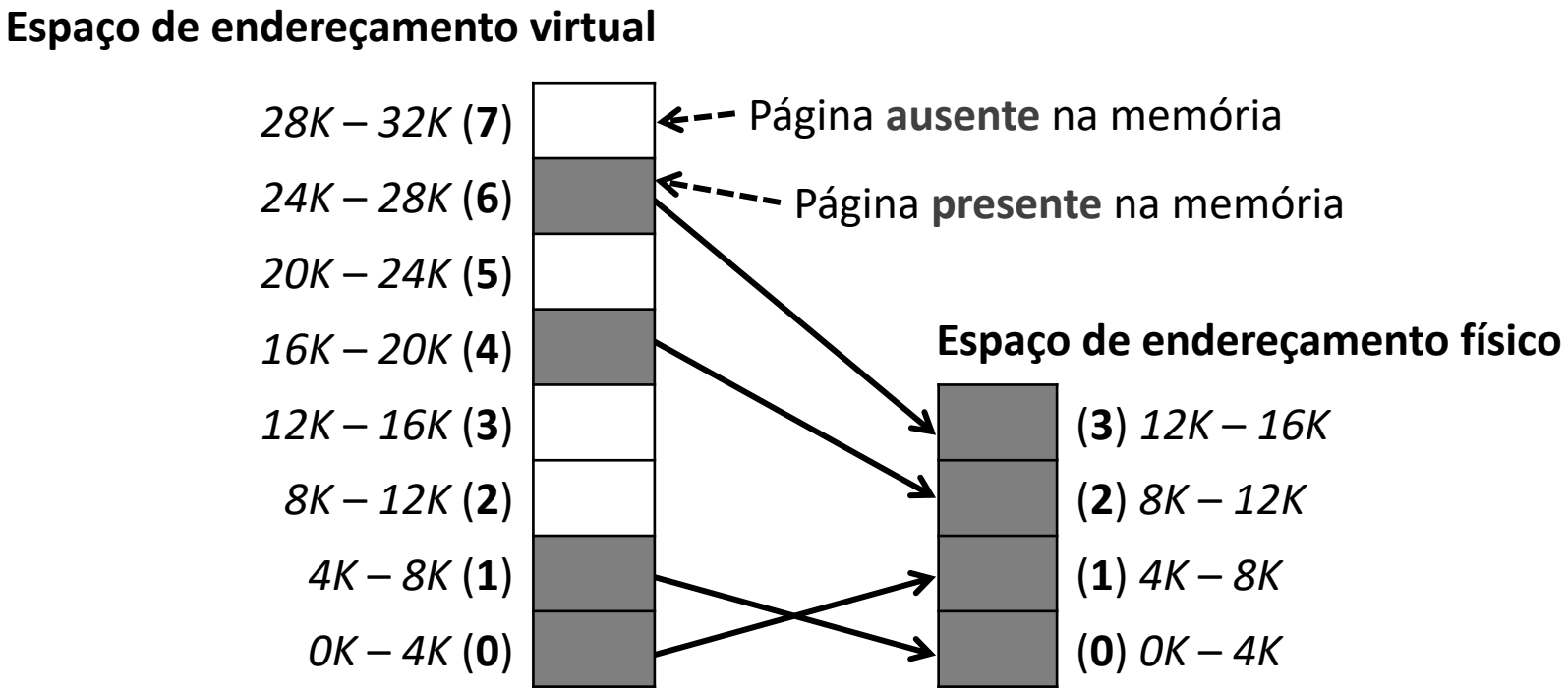
Memória virtual com paginação

- **Exemplo**: um sistema com endereços virtuais de **15 bits** (0 a 32K), **16 KB de memória física** e **páginas de 4 KB**



Memória virtual com paginação

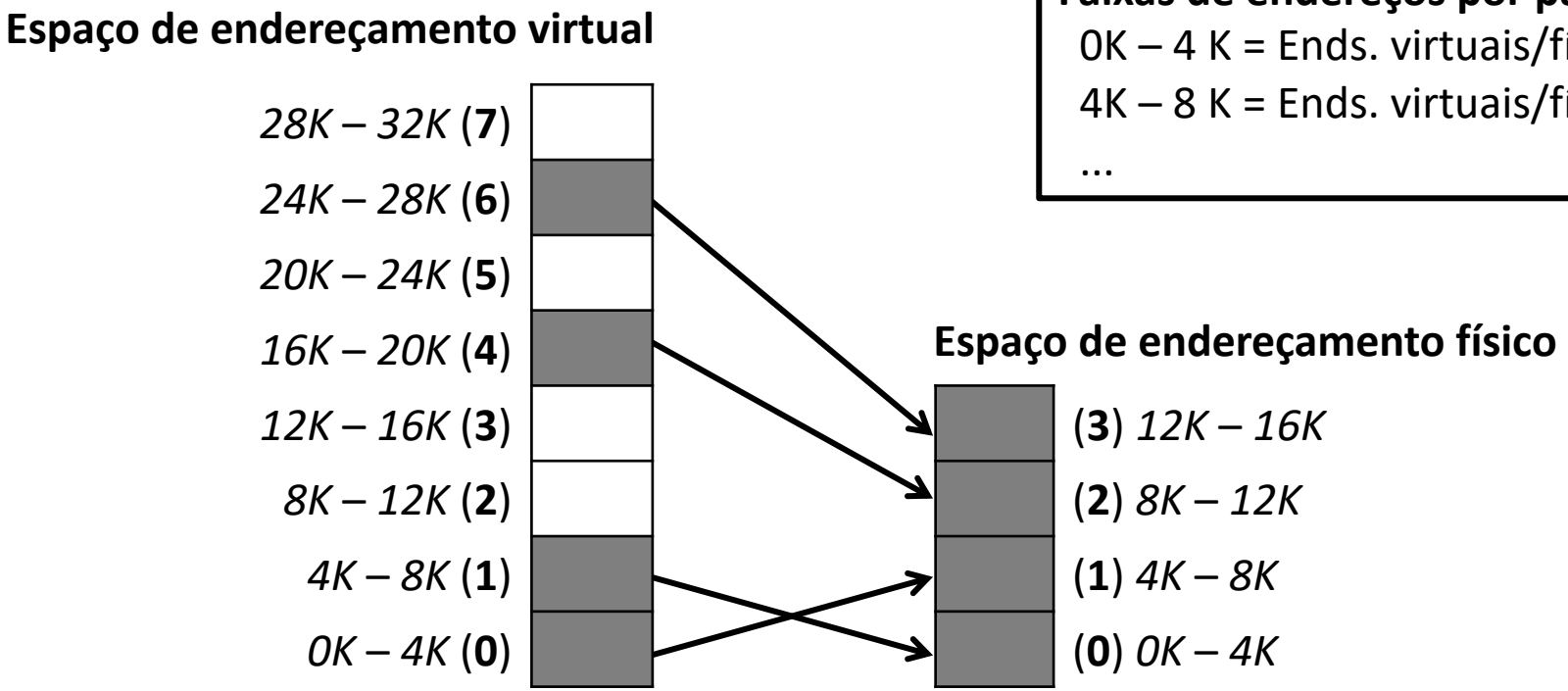
- Com **4 molduras de página**, somente **4 páginas** podem ser mapeadas para a memória física
- Um **bit presente/ausente** em cada **entrada da tabela de páginas** indica se a página está fisicamente presente na memória



Memória virtual com paginação

- Se o programa tenta acessar os seguintes **endereços virtuais**, quais serão os **endereços físicos** correspondentes?
  - 0000 = ?
  - 4098 = ?

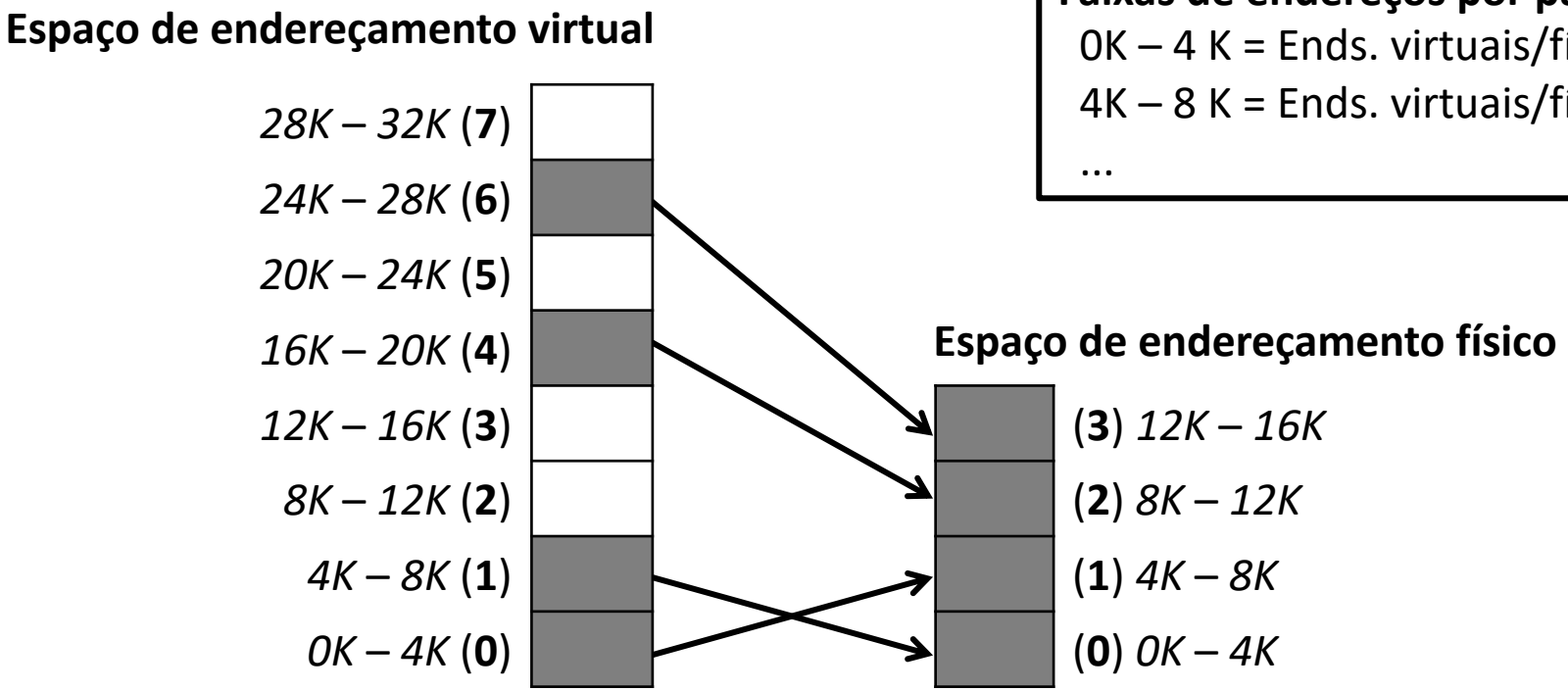
Faixas de endereços por página:
0K – 4 K = Ends. virtuais/físicos 0 a 4095
4K – 8 K = Ends. virtuais/físicos 4096 a 8191
...



Memória virtual com paginação

- Se o programa tenta acessar os seguintes **endereços virtuais**, quais serão os **endereços físicos** correspondentes?
  - 0000 = 4096 *(base)* + 0 *(deslocamento)* = **4096**
  - 4098 = ?

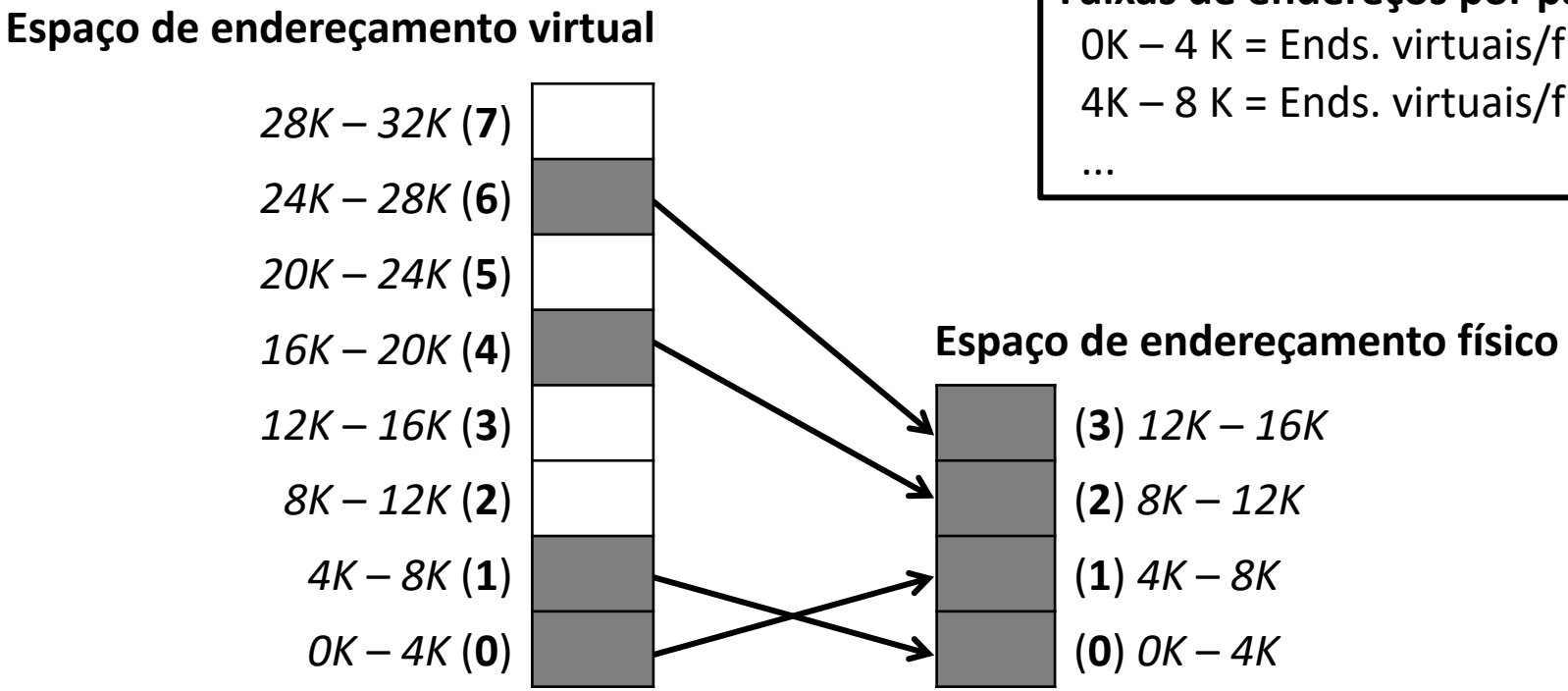
Faixas de endereços por página:
0K – 4 K = Ends. virtuais/físicos 0 a 4095
4K – 8 K = Ends. virtuais/físicos 4096 a 8191
...



Memória virtual com paginação

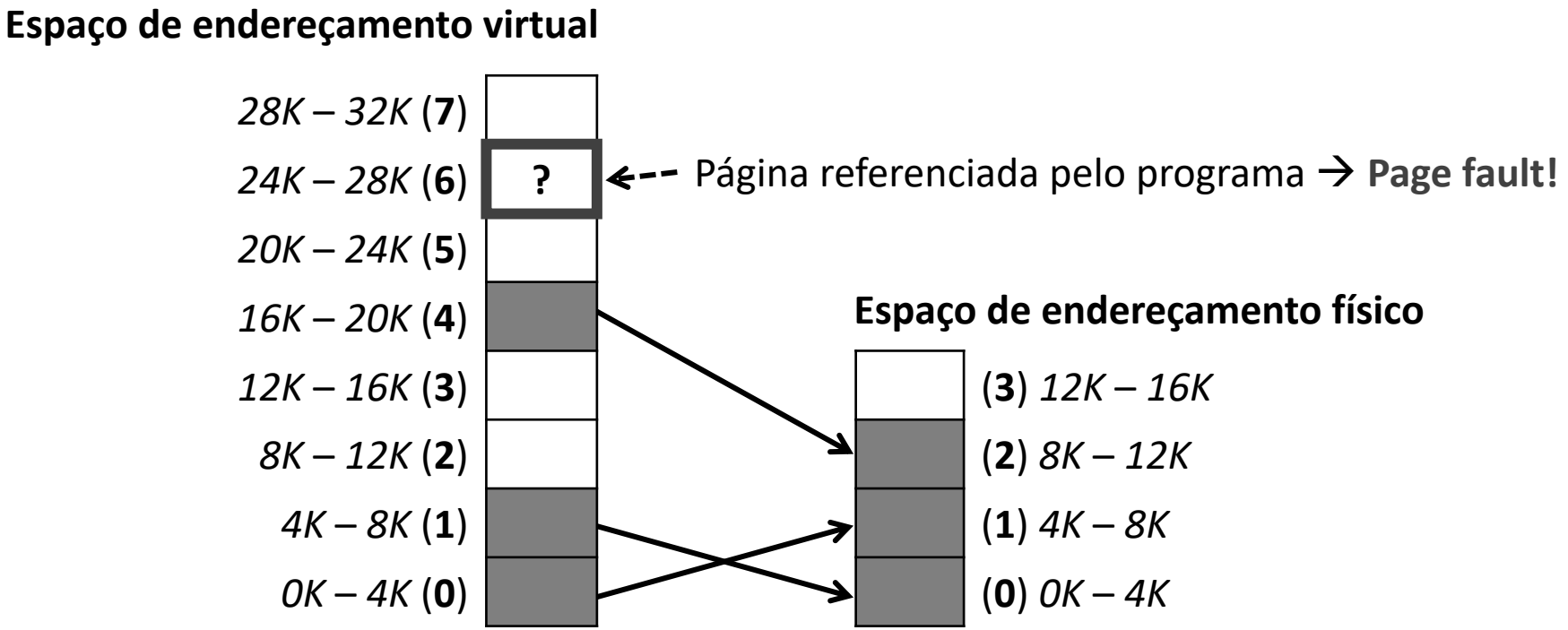
- Se o programa tenta acessar os seguintes **endereços virtuais**, quais serão os **endereços físicos** correspondentes?
  - 0000 = 4096 *(base)* + 0 *(deslocamento)* = **4096**
  - 4098 = 0 *(base)* + 2 *(deslocamento)* = **2**

Faixas de endereços por página:
0K – 4 K = Ends. virtuais/físicos 0 a 4095
4K – 8 K = Ends. virtuais/físicos 4096 a 8191
...



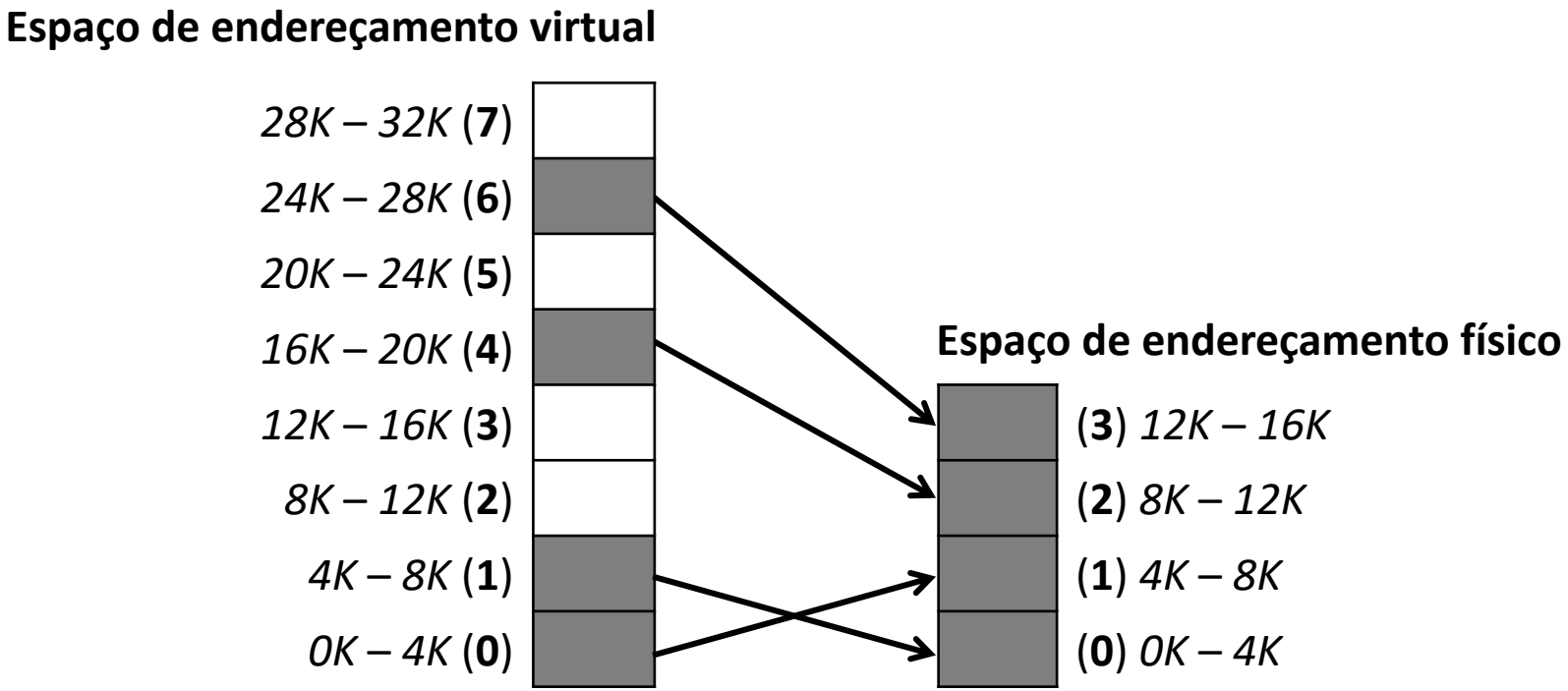
Memória virtual com paginação

- Quando uma página virtual é referenciada pelo programa e **não está mapeada na memória física** ocorre uma **falta de página** (***page fault***)
  - A página é **trazida para a memória** e o **mapeamento é atualizado**
  - A **instrução** que gerou a falta de página é **reexecutada**



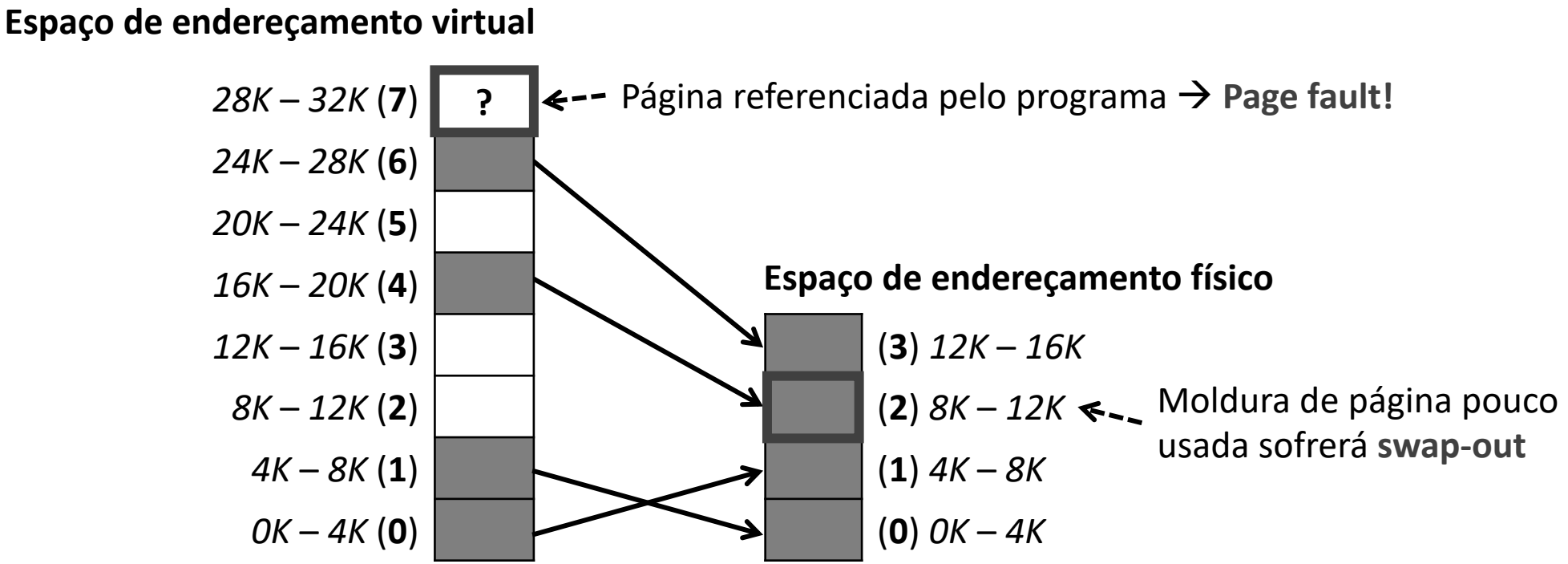
Memória virtual com paginação

- Quando uma página virtual é referenciada pelo programa e **não está mapeada na memória física** ocorre uma **falta de página** (***page fault***)
  - A página é **trazida para a memória** e o **mapeamento é atualizado**
  - A **instrução** que gerou a falta de página é **reexecutada**



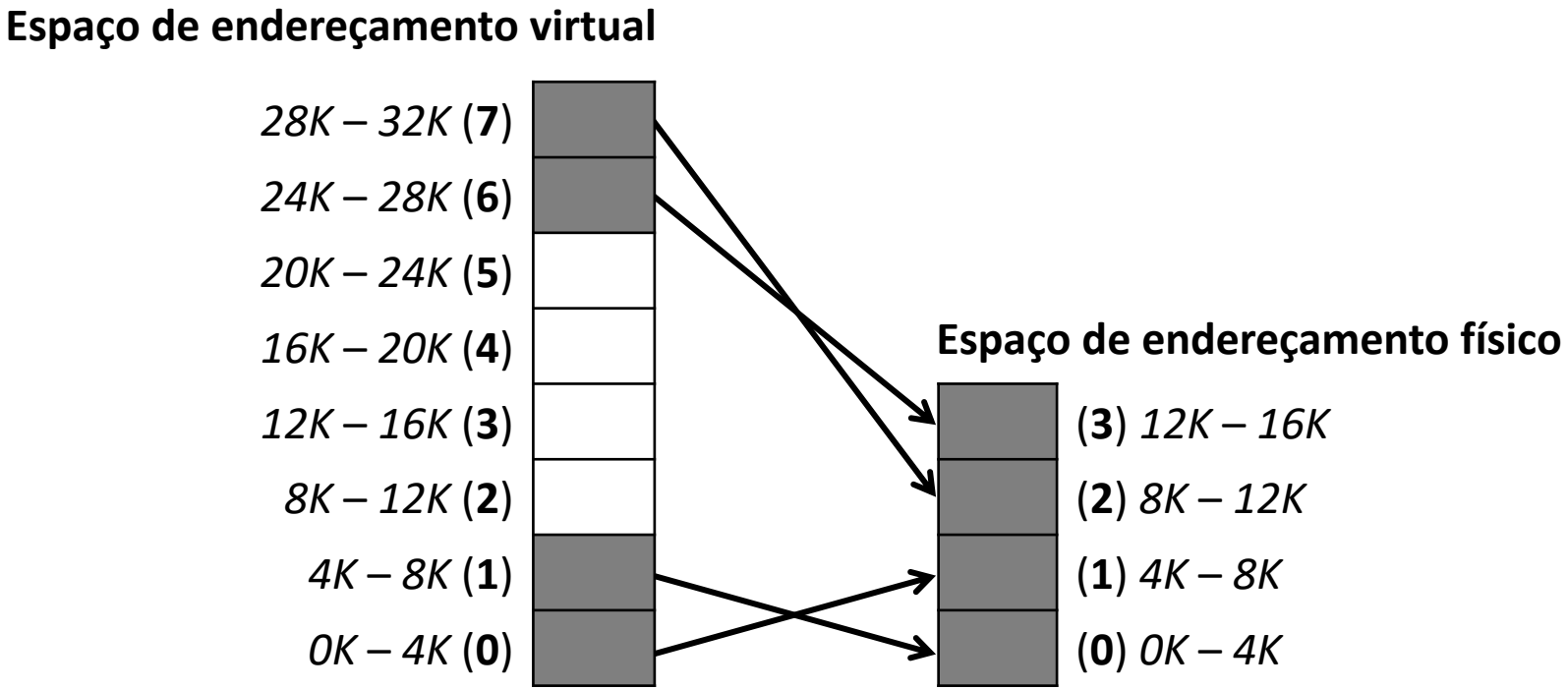
Memória virtual com paginação

- Quando ocorre uma **falta de página** e **não há moldura de página livre**
  - Uma moldura de página **pouco utilizada** sofrerá um **swap-out**
  - A página é **trazida para a memória** e o **mapeamento é atualizado**
  - A **instrução** que gerou a falta de página é **reexecutada**



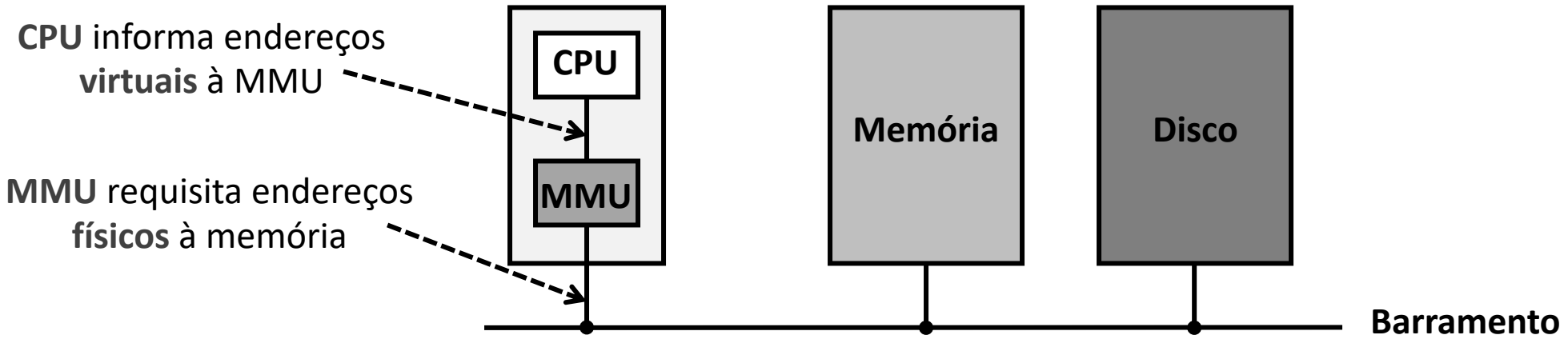
Memória virtual com paginação

- Quando ocorre uma **falta de página** e **não há moldura de página livre**
  - Uma moldura de página **pouco utilizada** sofrerá um **swap-out**
  - A página é **trazida para a memória** e o **mapeamento é atualizado**
  - A **instrução** que gerou a falta de página é **reexecutada**



Memória virtual com paginação

- **Memory Management Unit (MMU):** converte endereços virtuais em endereços físicos



Memória virtual com paginação

Exemplo

Endereços virtuais: 15 bits
Tamanho de páginas: 4 KB
Memória física: 16 KB

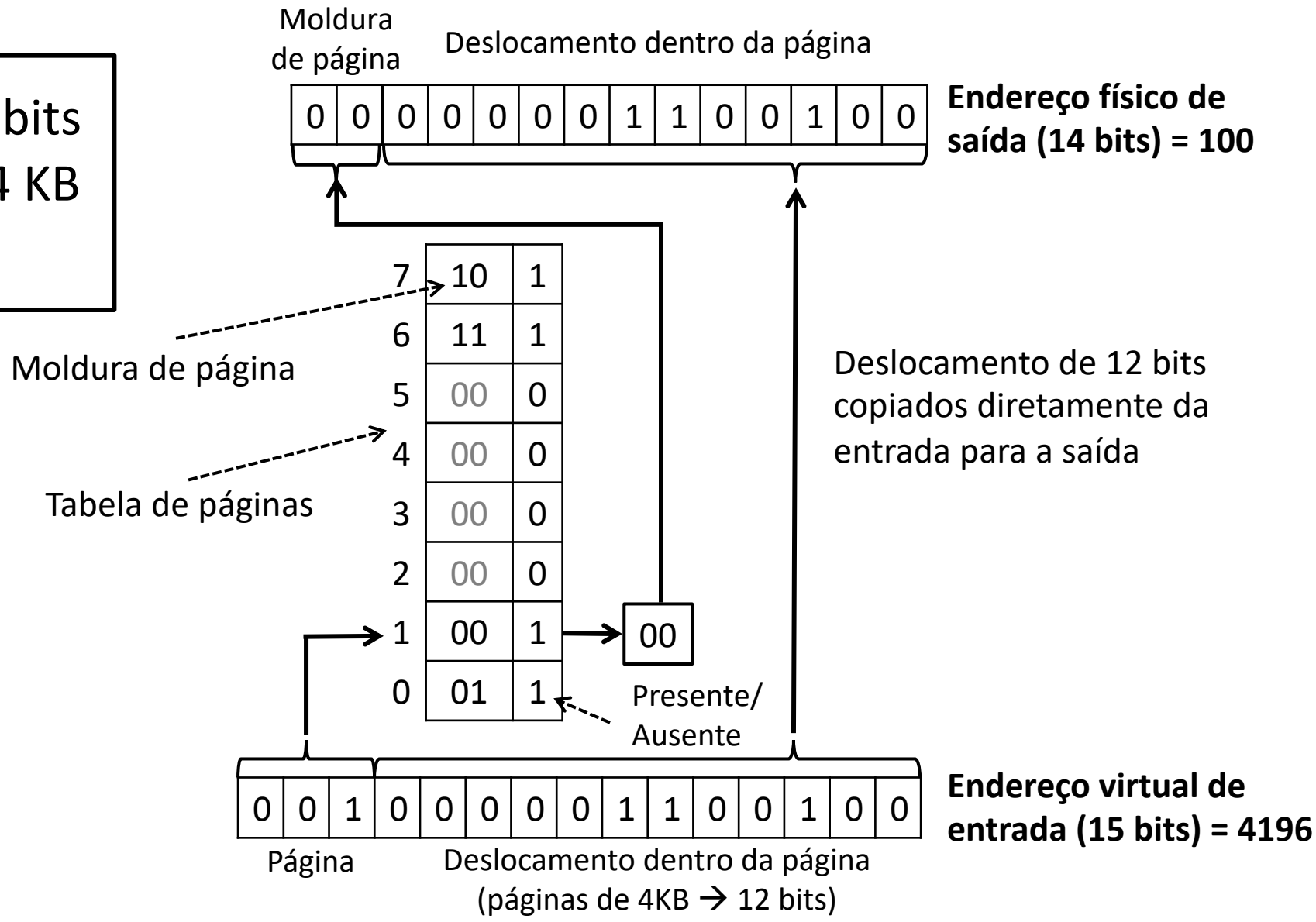


Tabela de páginas: componentes de cada entrada

- **Bit presente/ausente:** indica se está mapeada da memória física
- **Bits de proteção:** leitura, escrita ou ambos
- **Bit de modificada:** indica se foi modificada em memória
- **Bit referenciada:** indica se foi recentemente referenciada

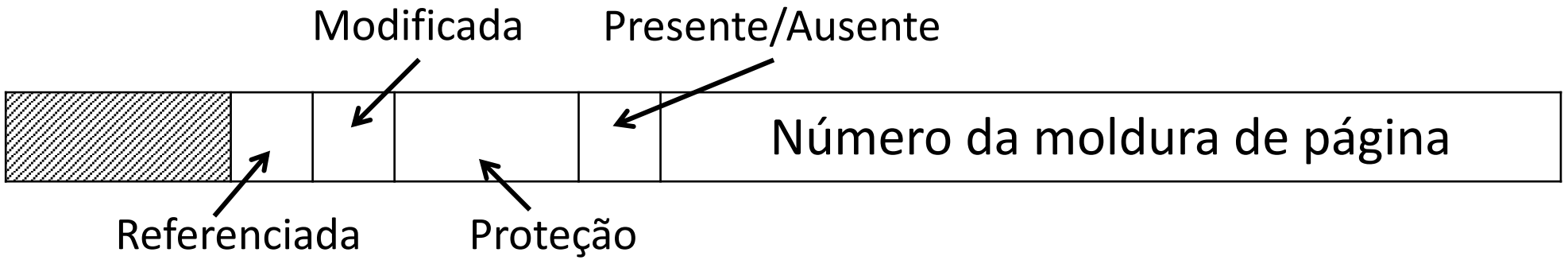


Tabela de páginas: armazenamento

- **A tabela de páginas é armazenada na memória RAM do computador**
  - Uma tabela de páginas **para cada processo**
- Registradores guardam as informações sobre os endereços da tabela
  - **PTBR**: endereço **base** da tabela de páginas
  - **PTLR**: endereço **limite** da tabela de páginas
- **PTBR e PTLR armazenados no descritor de processo**
  - Registradores são **atualizados** quando há **troca de contexto** entre processos

Tabela de páginas: armazenamento

Problema de armazenar a tabela de páginas na RAM

- Dois acessos à memória física a cada vez
  - Um acesso para **consultar a tabela de páginas** e fazer a tradução de endereço
  - Um acesso na **página requisitada**

Como evitar?

Translation Lookaside Buffer (TLB)

- ***Cache* em *hardware*** que **mapeia endereços virtuais em endereços físicos** sem passar pela tabela de páginas
  - Armazena páginas pertencentes ao ***working set* do programa**
  - Usa **memória associativa**
- **Conjunto de trabalho (*working set*)**
  - **Conjunto de páginas** mais prováveis de serem acessadas num dado momento, devido ao princípio de **localidade** (**espacial** e **temporal**)

TLB: Exemplo

Indica se a entrada é valida

Indica se a página foi modificada

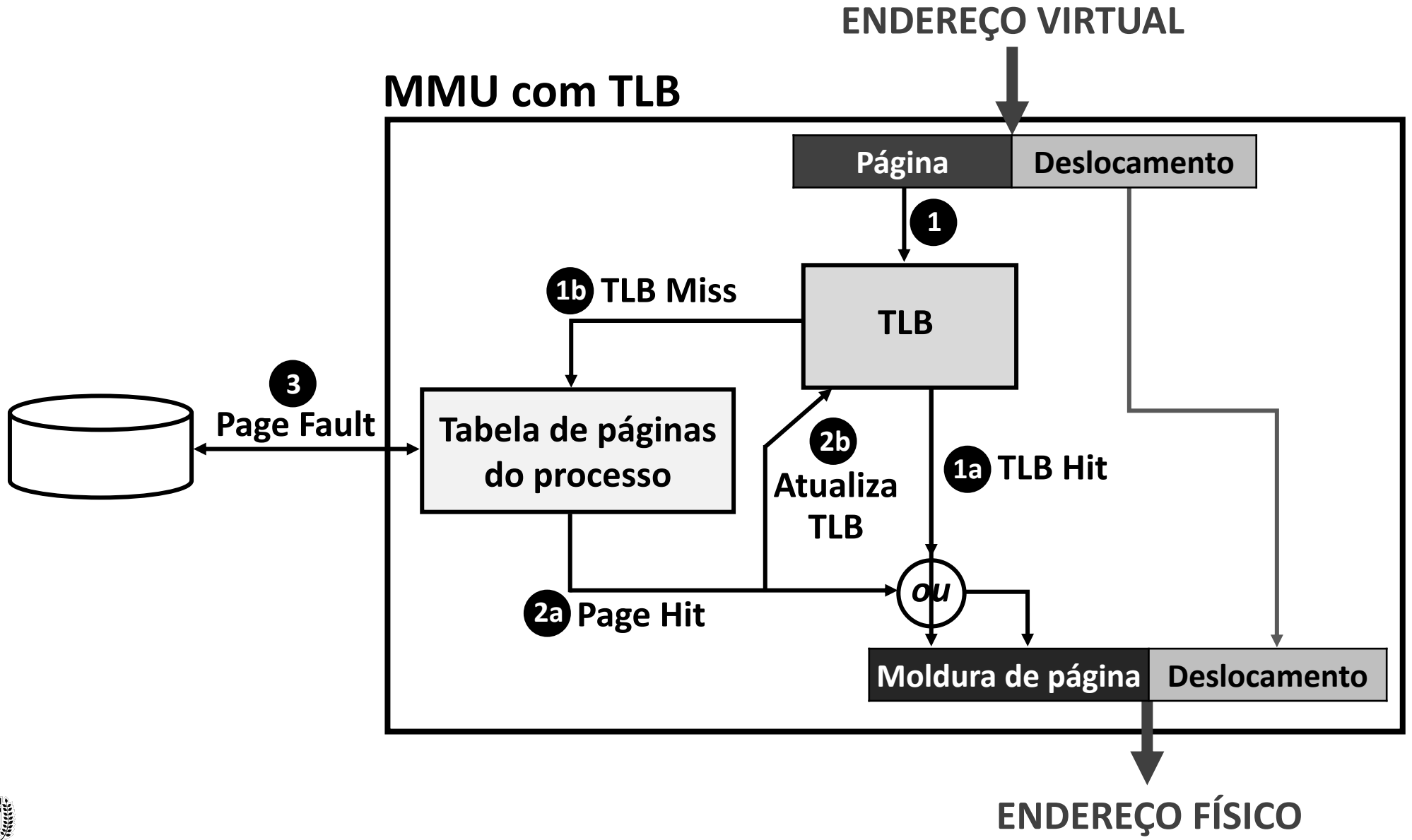
Permissões de leitura/escrita

Válida	Página	Modificada	Proteção	Moldura de página
1	120	1	0	12
1	20	0	1	31
1	180	1	1	20
0	134	0	0	55
1	43	0	1	104
0	33	0	1	75
1	513	1	0	42

TLB: Etapas da tradução de endereços

1. **Consultar a TLB usando os bits referentes à página do end. virtual**
   a. Se ***TLB hit***, a moldura de página é concatenada com o deslocamento, resultando no endereço físico a ser acessado na RAM
   b. Se ***TLB miss***, consulta a tabela de páginas
2. **Se a página está na tabela de páginas: *page hit***
   a. A moldura de página é concatenada com o deslocamento, resultando no endereço físico a ser acessado na RAM
   b. Atualiza a TLB para que as próximas traduções de endereço nessa página resultem em ***TLB hit***
3. **Se a página não está na tabela de páginas: *page fault***
   a. Realiza a operação de ***swap-in***
   b. Atualizada a tabela de páginas com o novo endereço virtual
   c. **Reinicializa a instrução** causadora do *page fault*

TLB: Etapas da tradução de endereços



Tabelas de páginas para memórias muito grandes

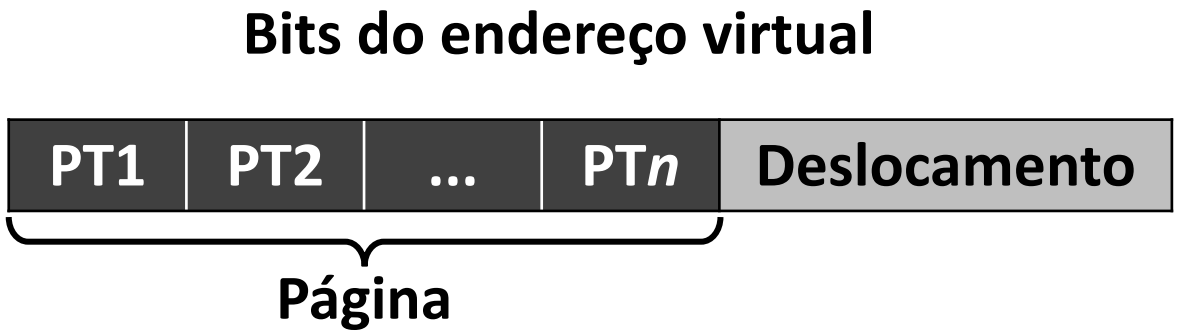
- Tabelas de páginas de são **mantidas na memória** em espaço de núcleo
  - **Uma tabela de páginas para cada processo**

- O tamanho da tabela de páginas será **muito grande** em sistemas que possuem um **grande espaço de endereçamento virtual**
  - **Exemplo:** endereços virtuais maiores que **32 bits**

- **Duas abordagens para lidar com esse problema**
  - Tabelas de páginas multinível
  - Tabelas de páginas invertidas

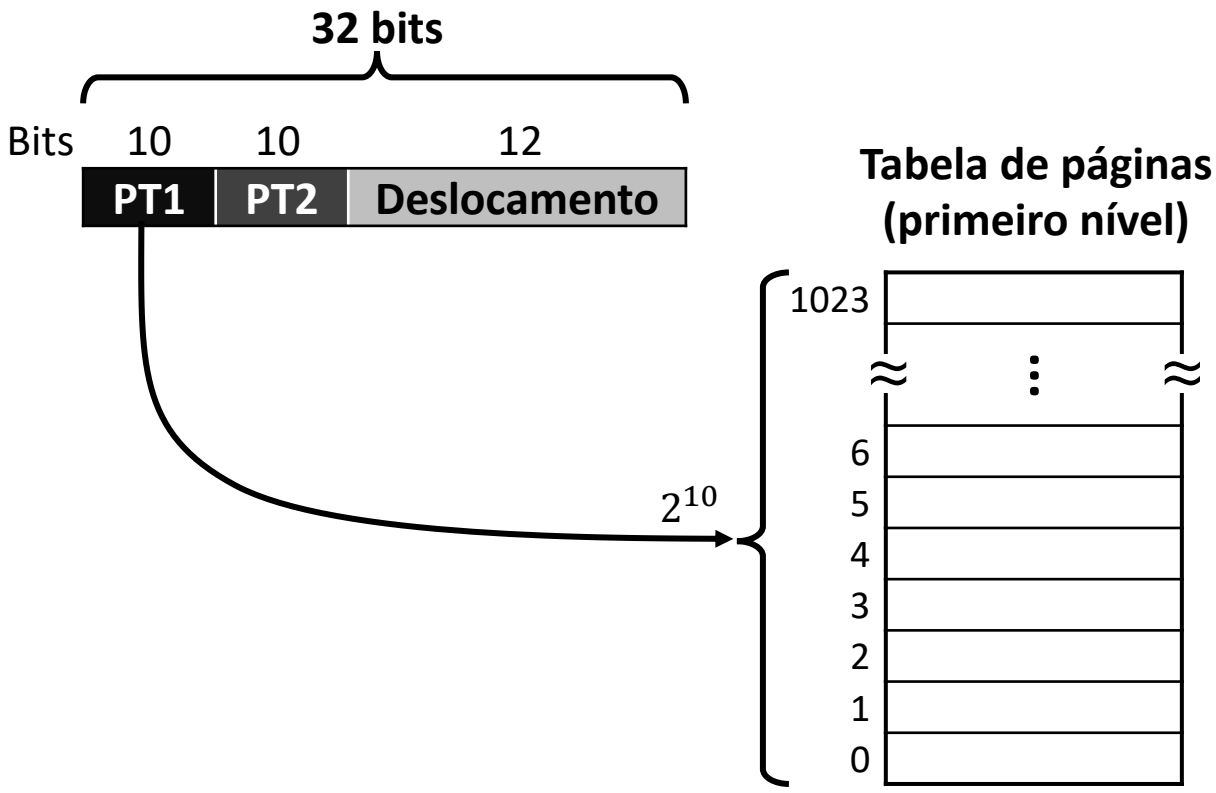
Tabelas de páginas multinível

- **Processos possuem mais de uma tabela de páginas**
  - Bits mais significativos do endereço virtual são **divididos em *n* níveis**
  - Todos os níveis, **exceto o último**, contêm **o endereço ou número da moldura de página de uma outra tabela de páginas**
  - **Último nível** contém o **número da moldura de página propriamente dita**

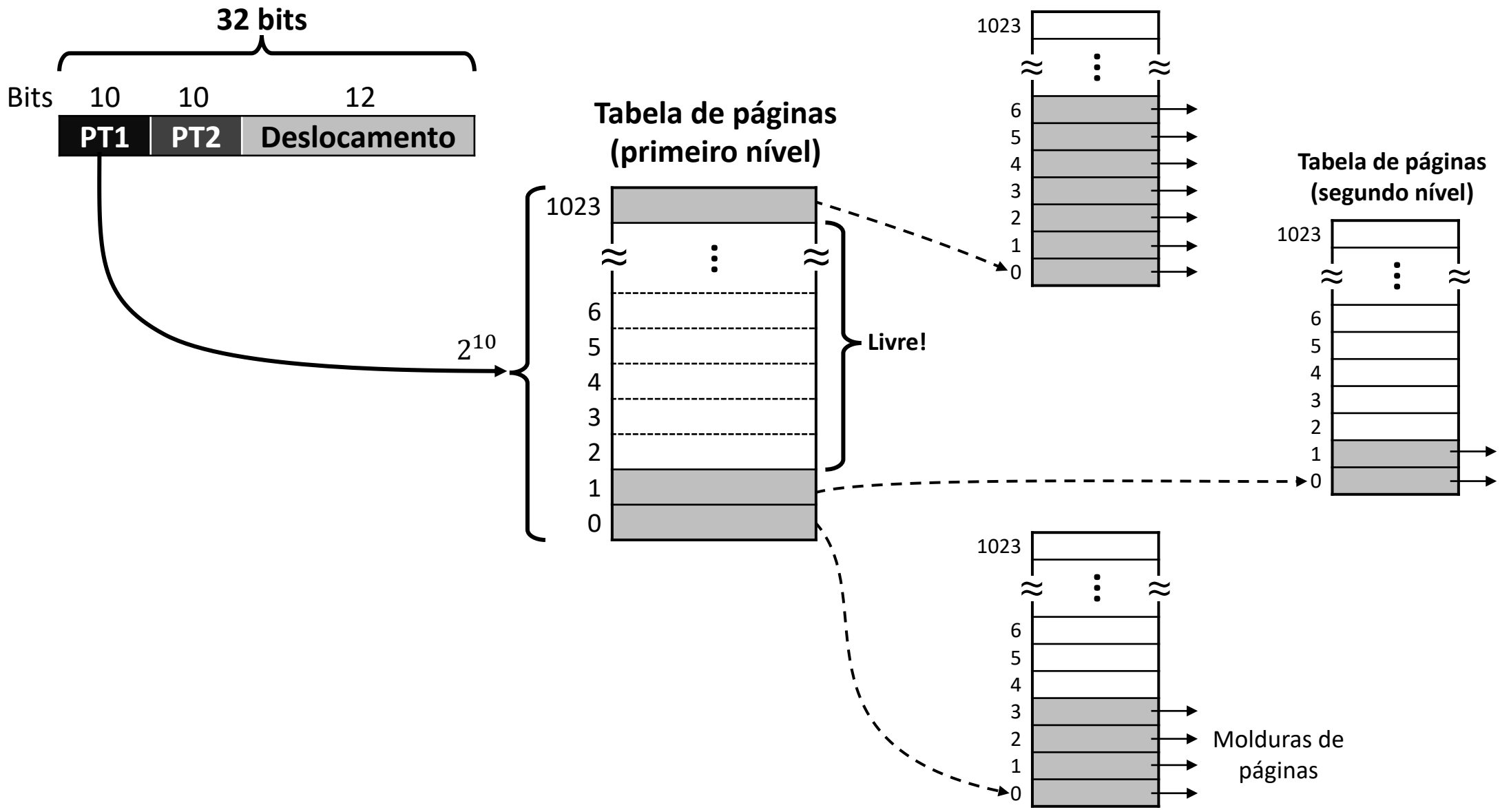


- **Evita** a necessidade de manter em memória **tabelas de páginas grandes**
  - Maioria dos processos **não usa** todo o seu espaço de endereçamento virtual

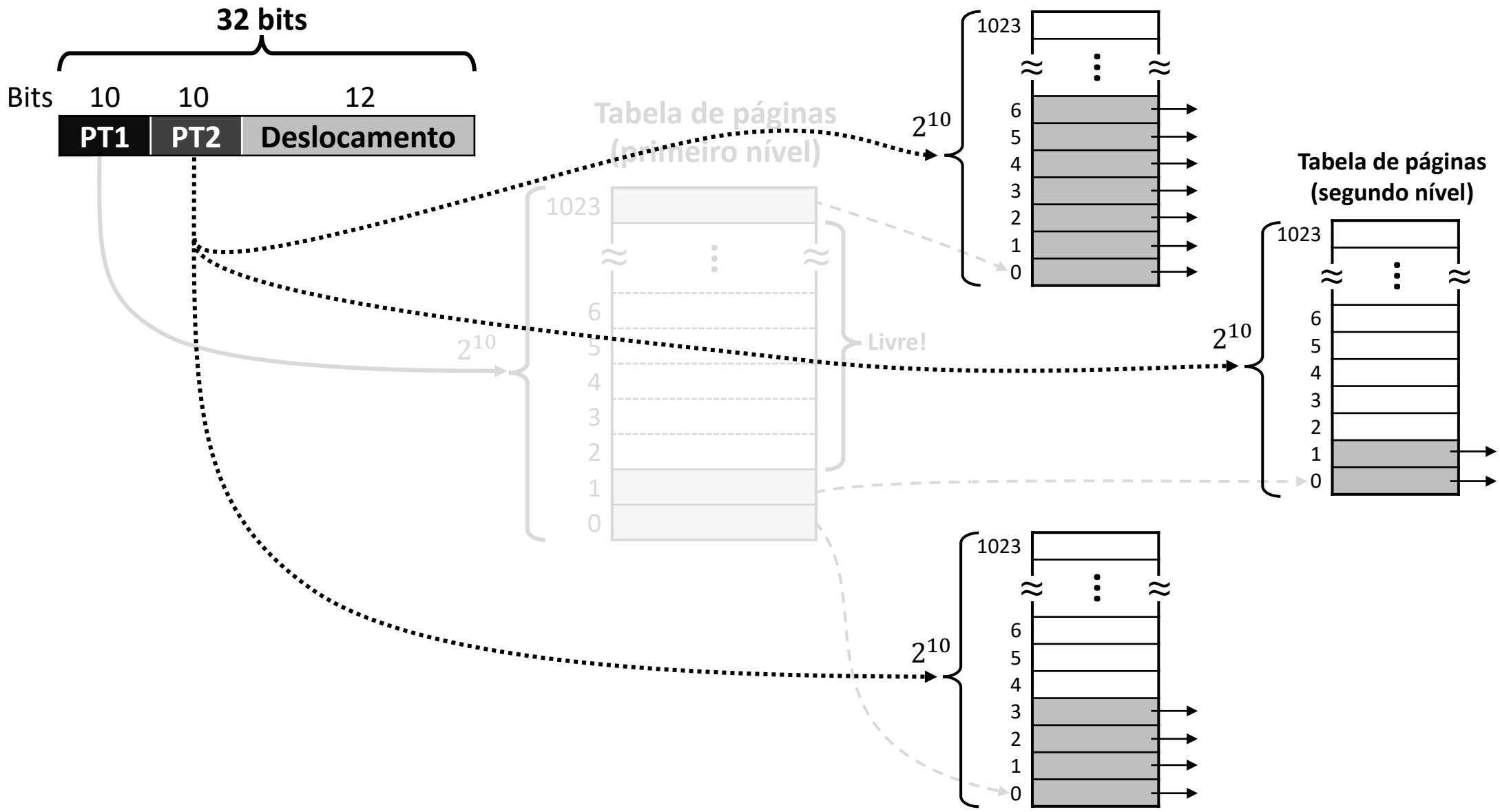
Exemplo: endereços de 32 bits, 2 níveis e páginas de 4 KB



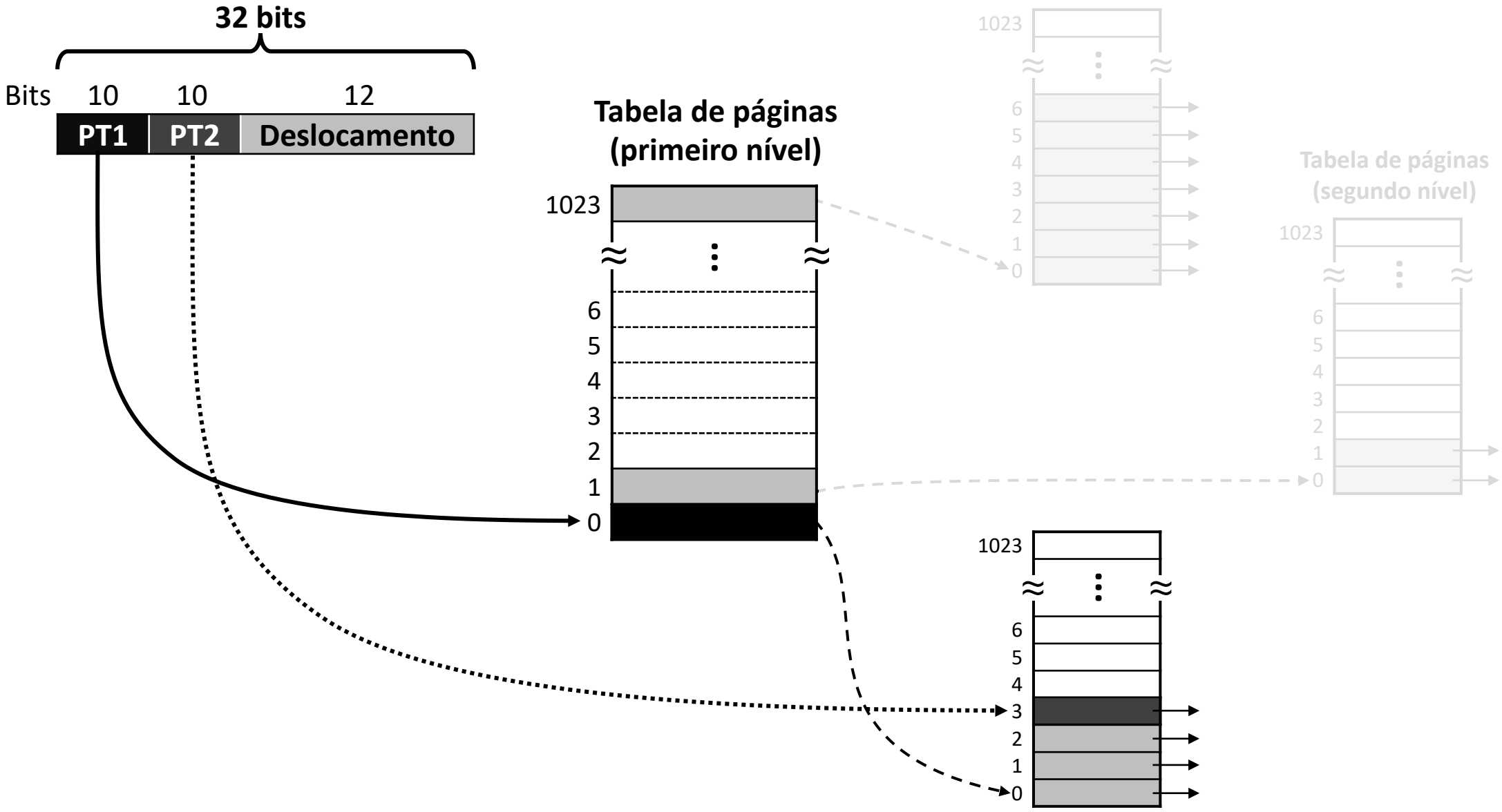
Exemplo: endereços de 32 bits, 2 níveis e páginas de 4 KB



Exemplo: endereços de 32 bits, 2 níveis e páginas de 4 KB



Exemplo: endereços de 32 bits, 2 níveis e páginas de 4 KB



Exemplo: endereços de 32 bits, 2 níveis e páginas de 4 KB

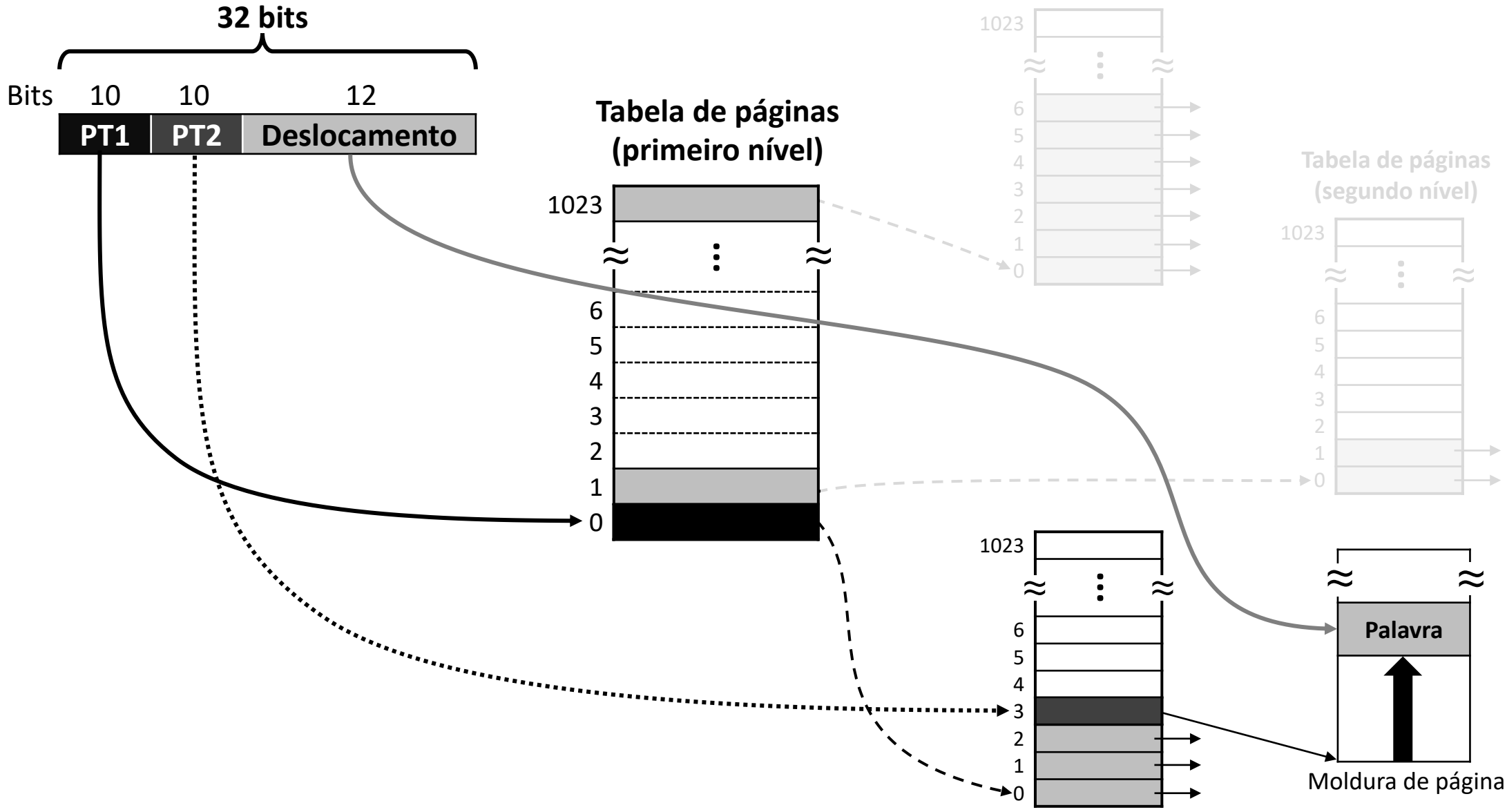


Tabela de páginas invertidas

- Uma **única tabela de páginas invertidas** para todos os processos, contendo em cada entrada:
  - **PID do processo**
  - **Número da página**
- **Número de entradas** da tabela de páginas invertidas = **número de molduras de página**
- **Índices** da tabela de páginas invertidas representam **números de molduras de página**

Molduras de página

	PID	Página virtual
0	6	100
1	2	232
2	10	432
3	32	120
4	1	56
5	13	323
n		

Tabela de páginas invertidas

Tabela de páginas invertidas

- **Tradução de endereços**
  - **PID do processo** e os bits referentes à **página** do endereço virtual são comparados com as entradas da tabela
  - Caso haja correspondência, o valor do **índice i** é utilizado nos **bits mais significativos do endereço físico (moldura de página)**
  - Bits de deslocamento **são copiados da entrada para a saída**

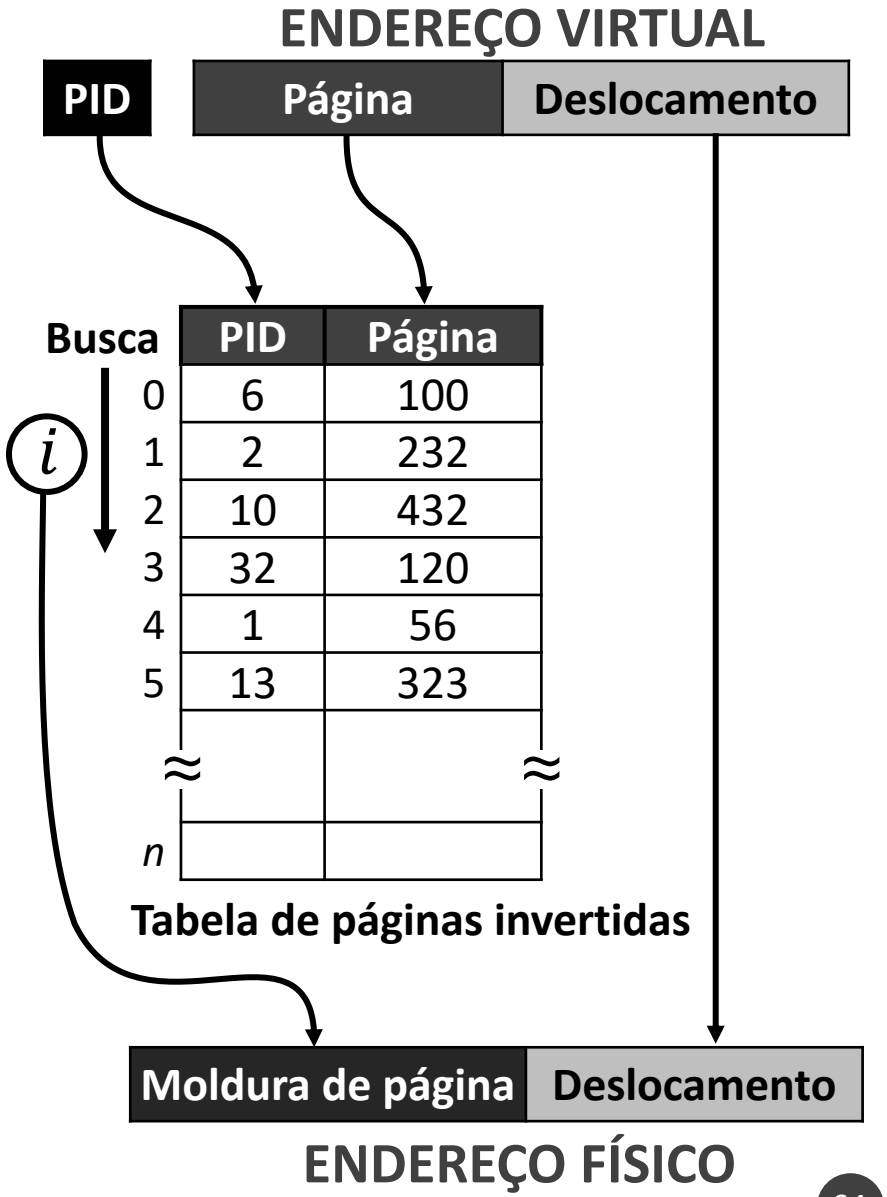


Tabela de páginas invertidas

- **Busca linear** na tabela de páginas invertidas apresenta um **péssimo desempenho**

- **Otimização:** ***Hashed Inverted Page Tables***
  - Utiliza uma **função *hash* `h(p, vpn)`** para encontrar a entrada na tabela de páginas invertidas que possa conter a **página `vpn`** de um **processo `p`**
  - Necessidade de **tratamento de colisões**

Obrigado pela atenção!

Dúvidas? Entre em contato:

- marcio.castro@ufsc.br
- www.marciocastro.com

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LaPeSD

Distributed Systems Research Lab

www.lapesd.inf.ufsc.br