Arquitetura de Computadores

MIEI – 2021/22

DI-FCT/UNL

Memoria Virtual

Sumário

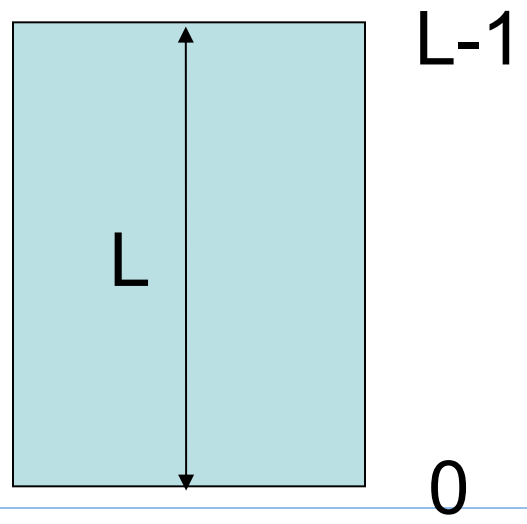
- Imagem (espaço de endereçamento) de um processo
- Endereços virtuais e físicos
- Gestão da Memória Virtual
  - Tradução de endereços
  - MMU
  - Paginação
  - Proteção da memória
  - TLB
  - Paginação a pedido

Bibliografia:

Bryant, O'Hallaron Computer Systems: A Programmer's Perspective, 2nd ou 3rd Ed, secções 9.1 a 9.6

Imagem do processo

- Conteúdo das zonas de memória que pertencem ao processo
- Constituído por um conjunto de endereços contíguos de L bytes de comprimento
- Pode-se endereçar qualquer byte entre 0 e L-1



Organização da memória física

- O processo vai ser carregado em diferentes posições da memória física
- Só no momento do carregamento se sabe qual a memória que está livre

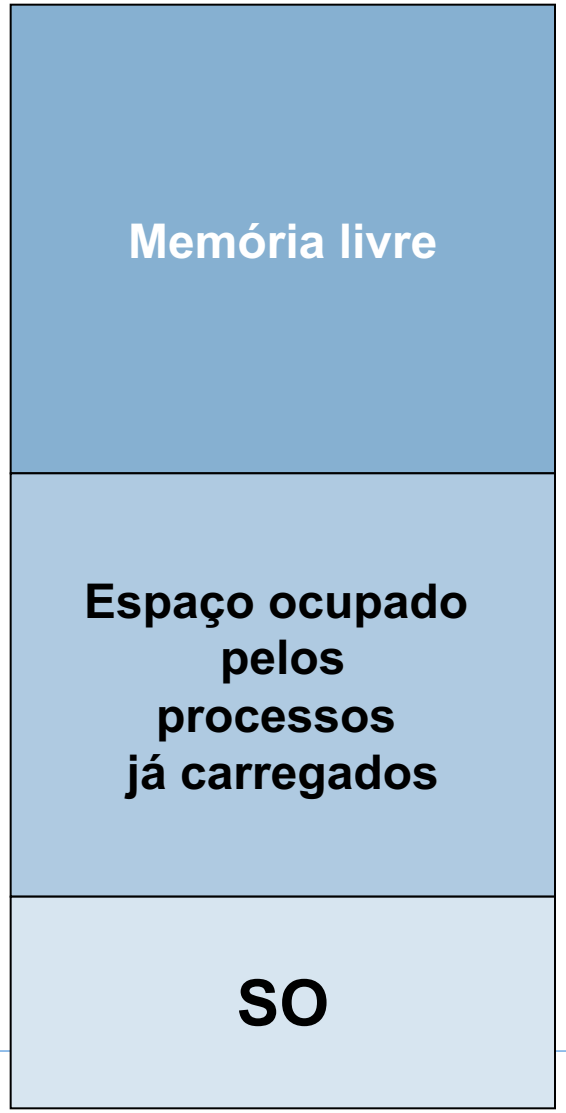


Imagem do processo

Imagem de memória de um processo

Linux/x86

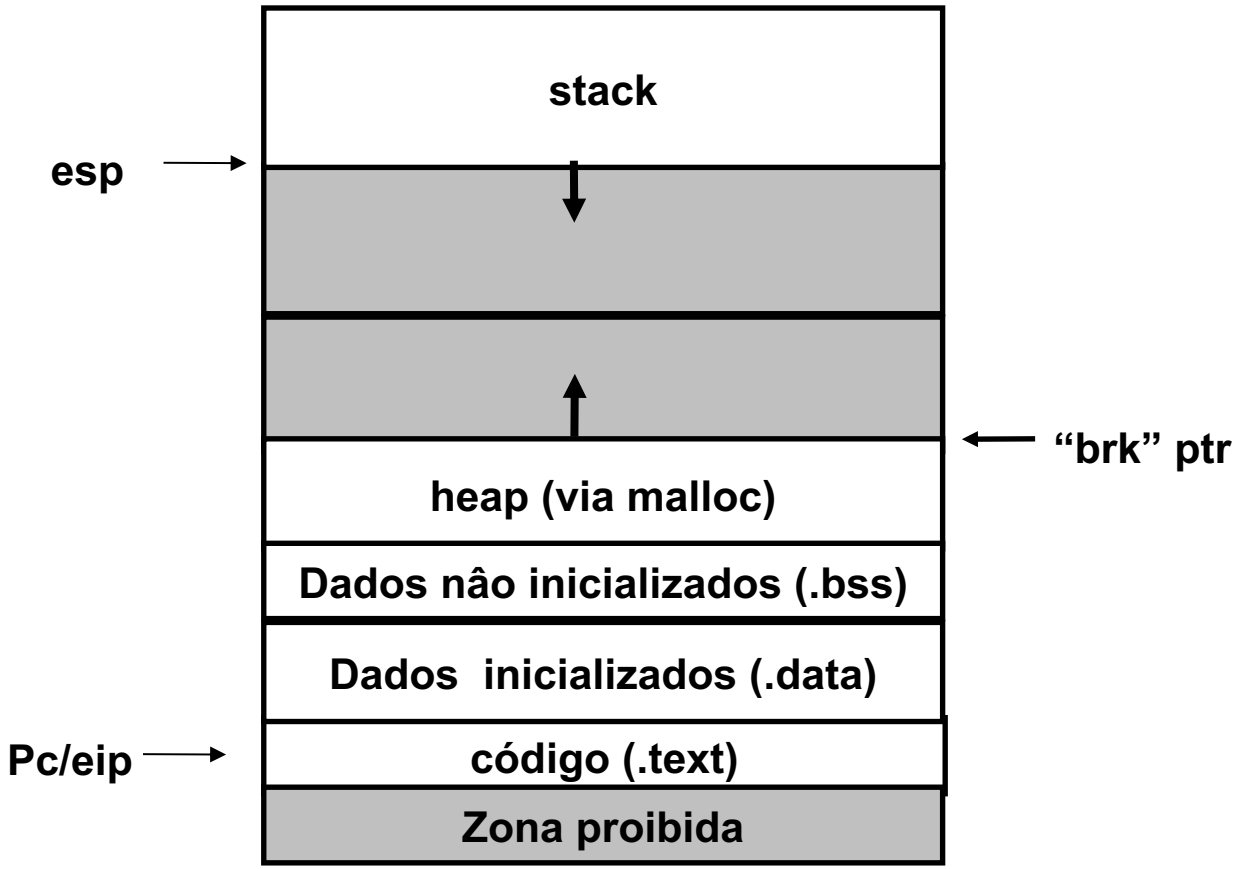
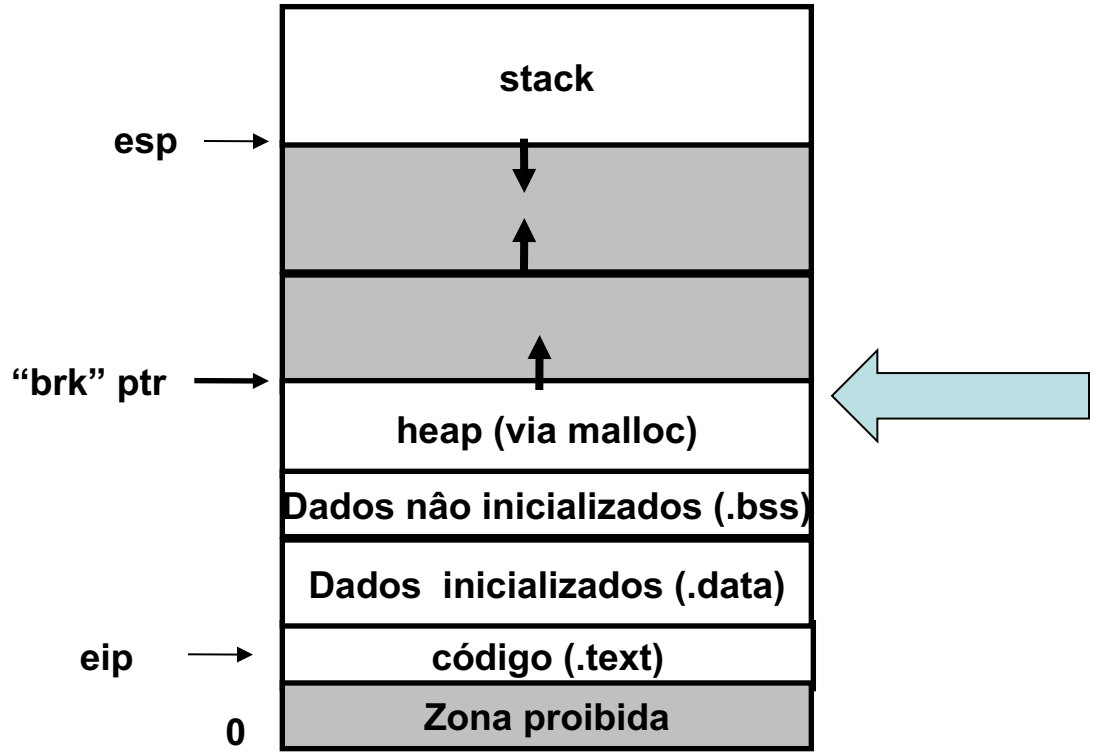


Imagem do processo

Operações que fazem variar o tamanho do heap



Crescimento

*void *malloc(size_t size)*

Decrescimento

*void free(void *ptr)*

Espaços de endereços lógicos (ou virtuais) e físicos

- Em cada momento, estão carregados em memória imagens (código + dados + pilha) de vários processos
- Cada processo tem um espaço de endereçamento lógico que é separado do dos outros processos
- O conceito de um *espaço de endereçamento lógico* é independente do *espaço de endereços físico*
  - ***Endereços lógicos*** – gerados pelo CPU; também conhecidos por ***endereços virtuais***.
  - ***Endereços físicos*** – endereços recebidos pela memória física

- **Os endereços virtuais e físicos diferem.**

Unidade de gestão de memória (MMU – memory management unit)

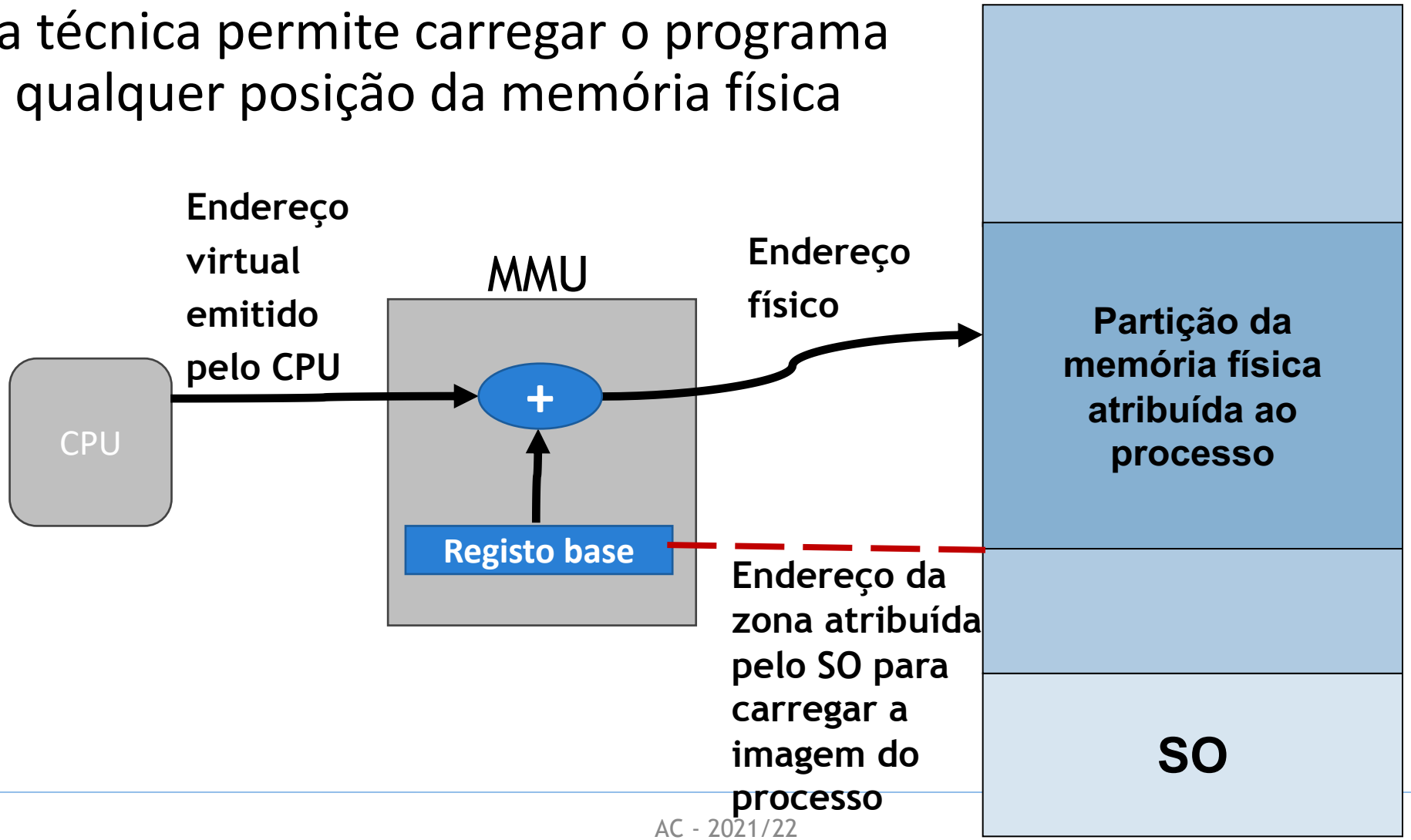
- Dispositivo hardware que faz a correspondência (mapping) entre endereços virtuais e físicos.

- **O programa gera endereços *virtuais***
  - não controla os endereços físicos.

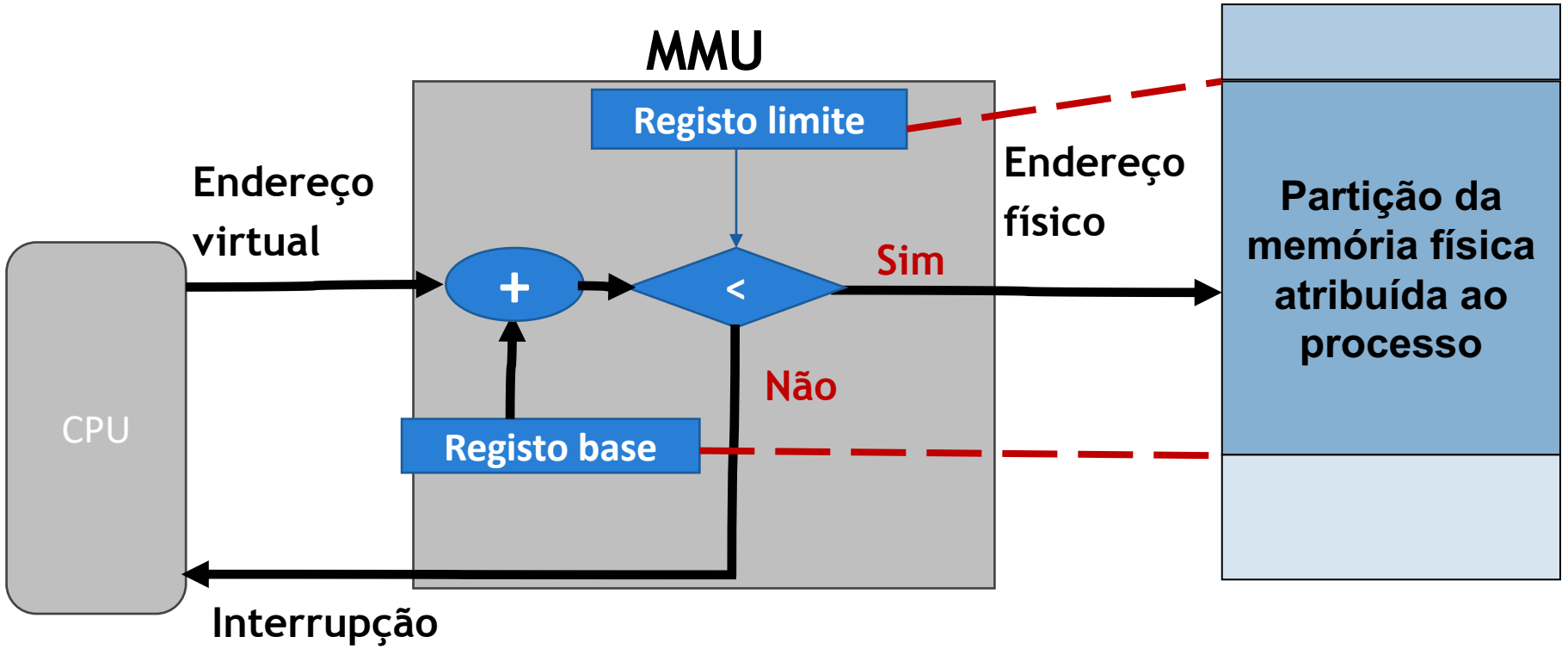
- As MMUs mais simples usam um ***registo de recolocação***: o valor deste registo é somado a todos os endereços gerados pelo programa antes de estes serem enviados para a memória.

MMU com registo de recolocação

- Esta técnica permite carregar o programa em qualquer posição da memória física



MMU com registo de recolocação (registo base) e registo limite



- Este hardware permite não só a recolocação da imagem dos processos, como impede que um processo tenha acesso às imagens de outros processos

Espaço físico gerido por alocação contígua

- Memória dividida em duas zonas:
  - SO.
  - Espaço para imagens de processos.

- Espaço do utilizador
  - Dividido em partições: zonas contíguas na memória física

Espaço físico gerido por alocação contígua

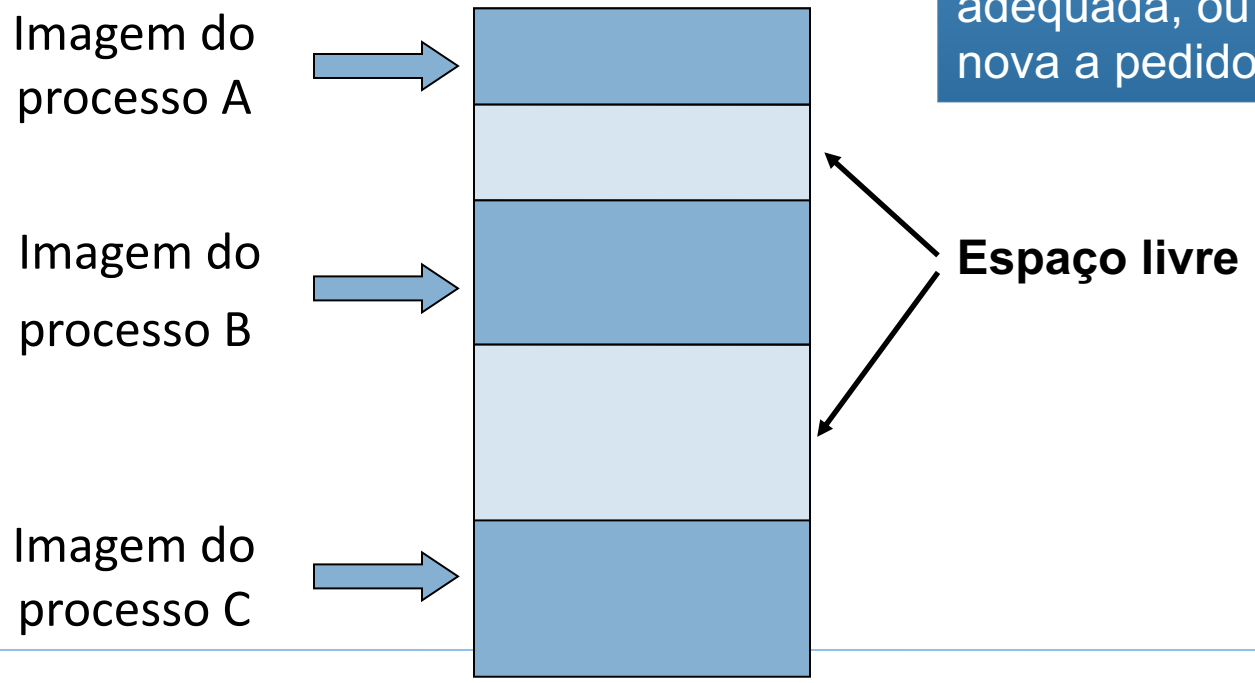
- Cada imagem de um processo tem de ser carregada numa partição de dimensão maior ou igual à dimensão da imagem do processo

- Suporte hardware: registos base e limite
  - Asseguram a recolocação do programa e a proteção
  - O valor corrente dos registos base e limite estão associados ao processo

Espaço físico gerido por alocação contígua

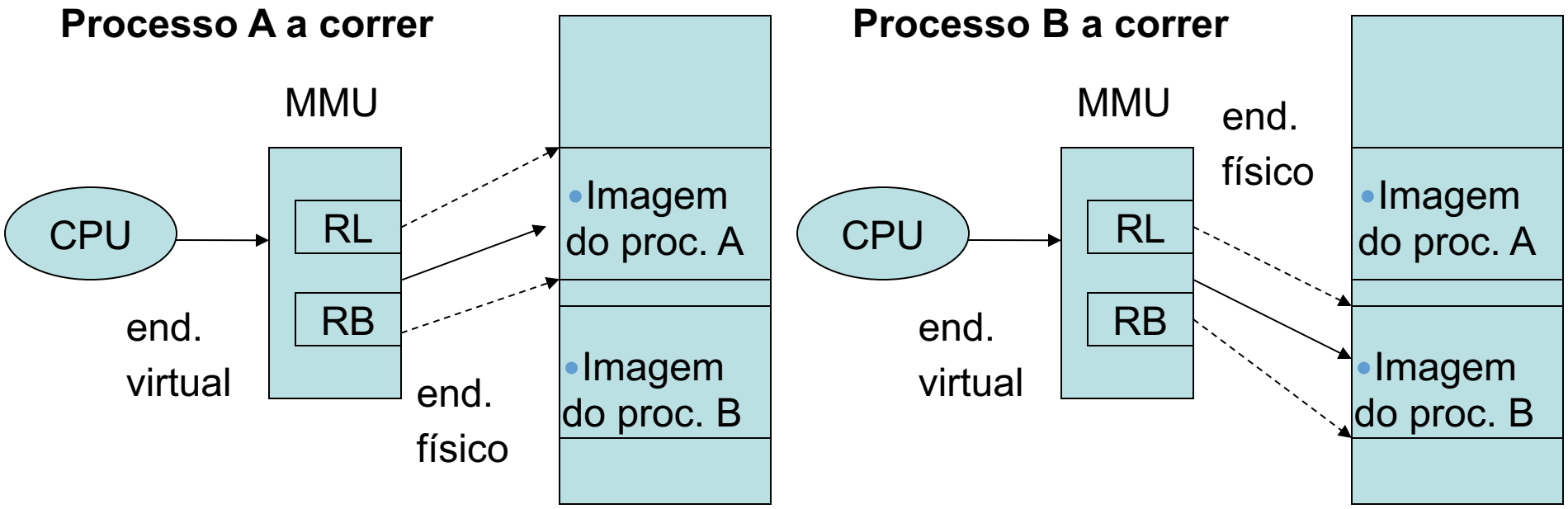
- Cada imagem de processo tem de ser carregada numa partição de dimensão maior ou igual à dimensão da imagem do processo

Para carregar um processo tem de estar livre uma partição de dimensão adequada, ou então ser criada uma nova a pedido na altura



Hardware necessário: registos base e limite

- Suporte hardware: registos base e limite
  - Asseguram a recolocação do programa e a protecção
  - O valor corrente dos registos base (RB) e limite (RL) são guardados no descritor do processo



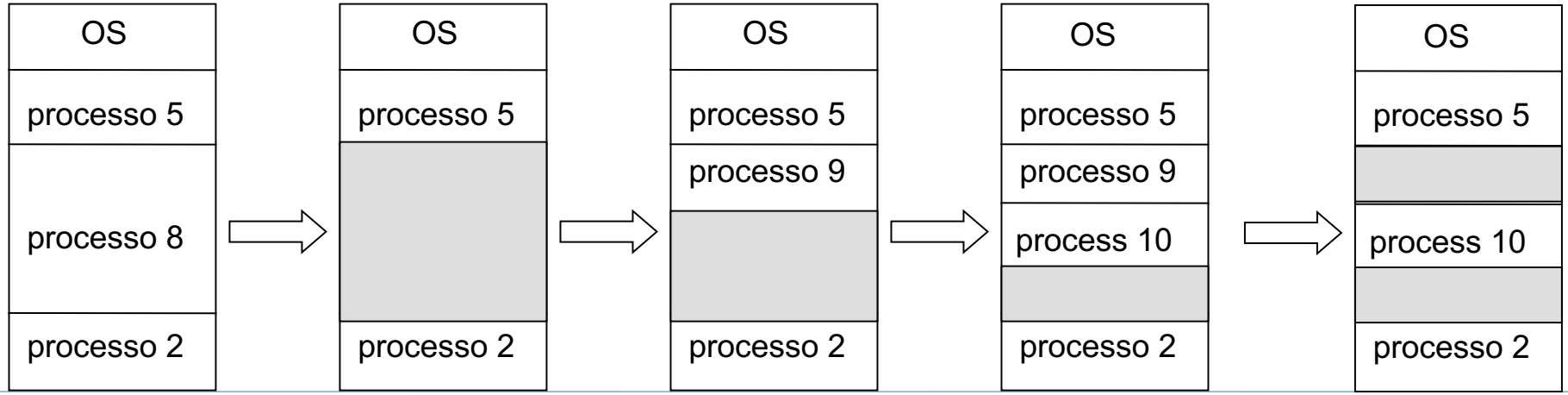
Formas de organizar as partições

- Número de partições fixo e de dimensão fixa
  - **pouco flexível**
    - apenas utilizável em ambientes em que as necessidades variam pouco
  - nº de processos máximo igual ao nº de partições
  - dado um processo cuja imagem tem dimensão, L se não houver uma partição de dimensão igual ou superior livre, o processo não pode ser carregado

- Número de partições variável e com dimensão também variável
  - existe um conjunto de memória livre e a partição é criada “à medida” do tamanho da imagem do processo a carregar

Inconvenientes da atribuição de memória por partições contíguas

- Zonas livres de diferentes dimensões estão espalhadas pela memória física.
- Quando um processo é criado, é necessário atribuir-lhe uma zona livre contígua suficientemente grande
- Ao fim de muitas operações de criação e destruição de processos, a memória livre fica fragmentada



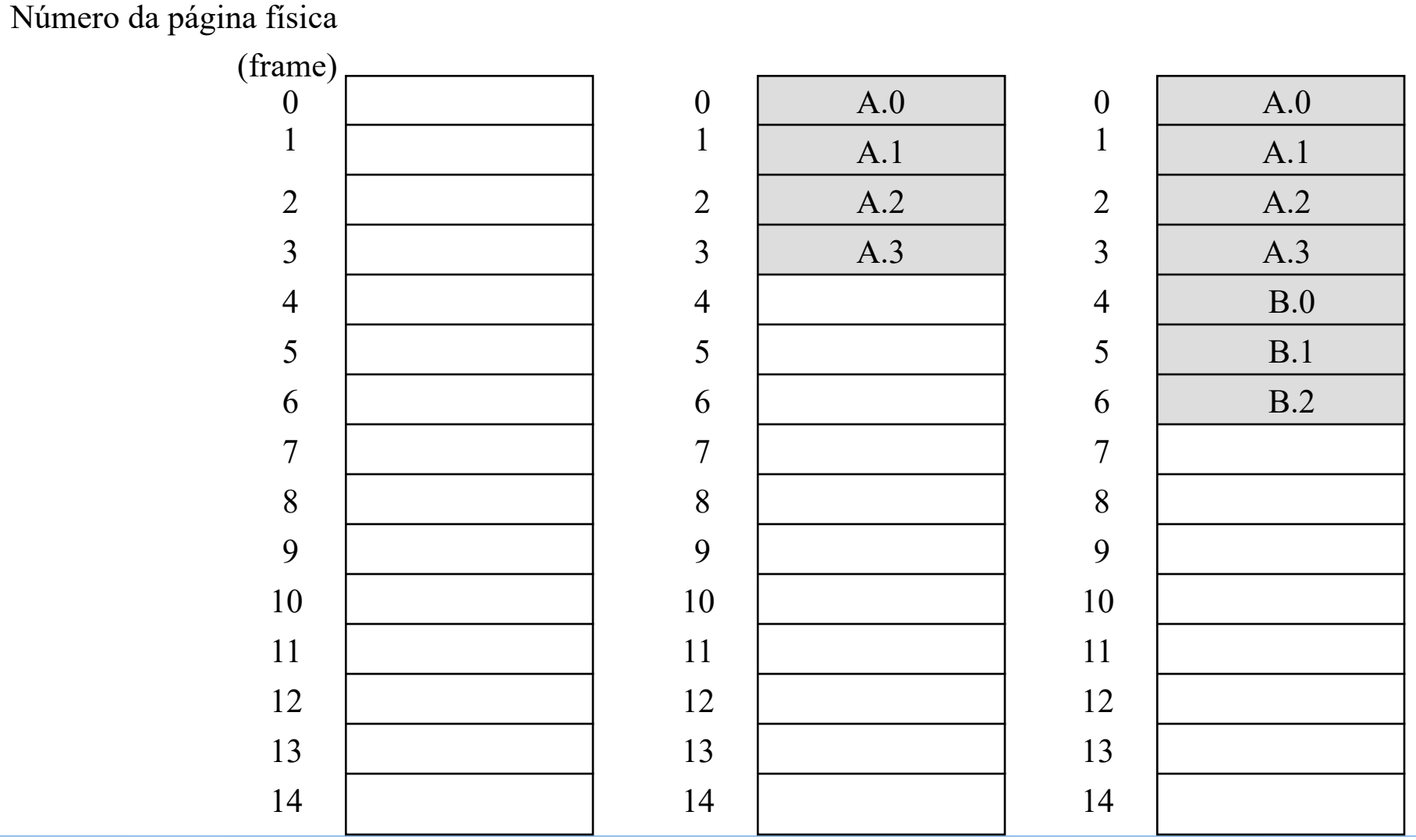
Fragmentação

- **Fragmentação externa** – existe memória física livre em quantidade suficiente para carregar o processo, mas não está contígua.
- **Fragmentação interna** – espaço existente dentro de uma partição que não é usado porque a a dimensão da imagem é inferior ao da partição
- Operações de compactação reduzem o problema da fragmentação externa
  - Juntar a memória livre num bloco único.
  - Operação feita em tempo de execução; só possível com recolocação dos programas (ex: registos base e limite)
  - Operação que exige muitas operações de I/O e provoca a paragem temporária de todos os programas envolvidos (pode requerer escritas em disco - **swapping**)

Paginação

- O espaço de endereçamento de um processo não necessita de ser contíguo – atribui-se memória física ao processo onde quer que ela esteja disponível
- Divide-se a memória física em blocos de tamanho fixo chamadas páginas físicas (**frames**) – tamanho é uma potência de 2 entre 512 bytes e 8192 bytes.
- Divide-se o espaço de endereçamento lógico em blocos do mesmo tamanho chamadas **páginas** (ou páginas virtuais).
- Manter informação sobre todas as frames livres.
- Para executar um programa com *n* páginas, é preciso encontrar *n* frames livres e carregar lá o programa.
- Preparar um tabela de páginas que traduz endereços virtuais em endereços físicos.
- Não há fragmentação externa
- Há fragmentação interna (em média ½ página)

Carregando os processos A e B em memória



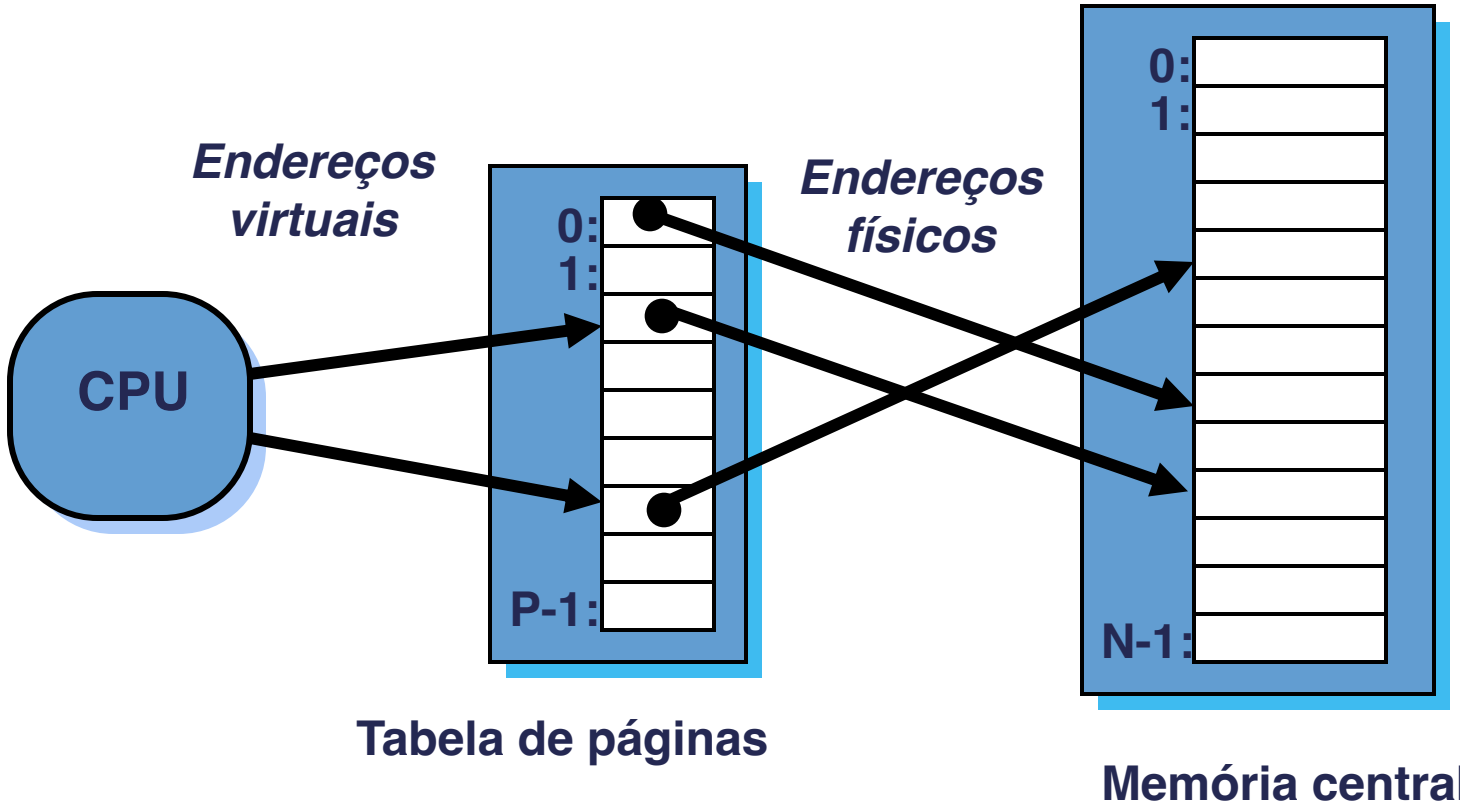
Paginação: atribuição não contígua

0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

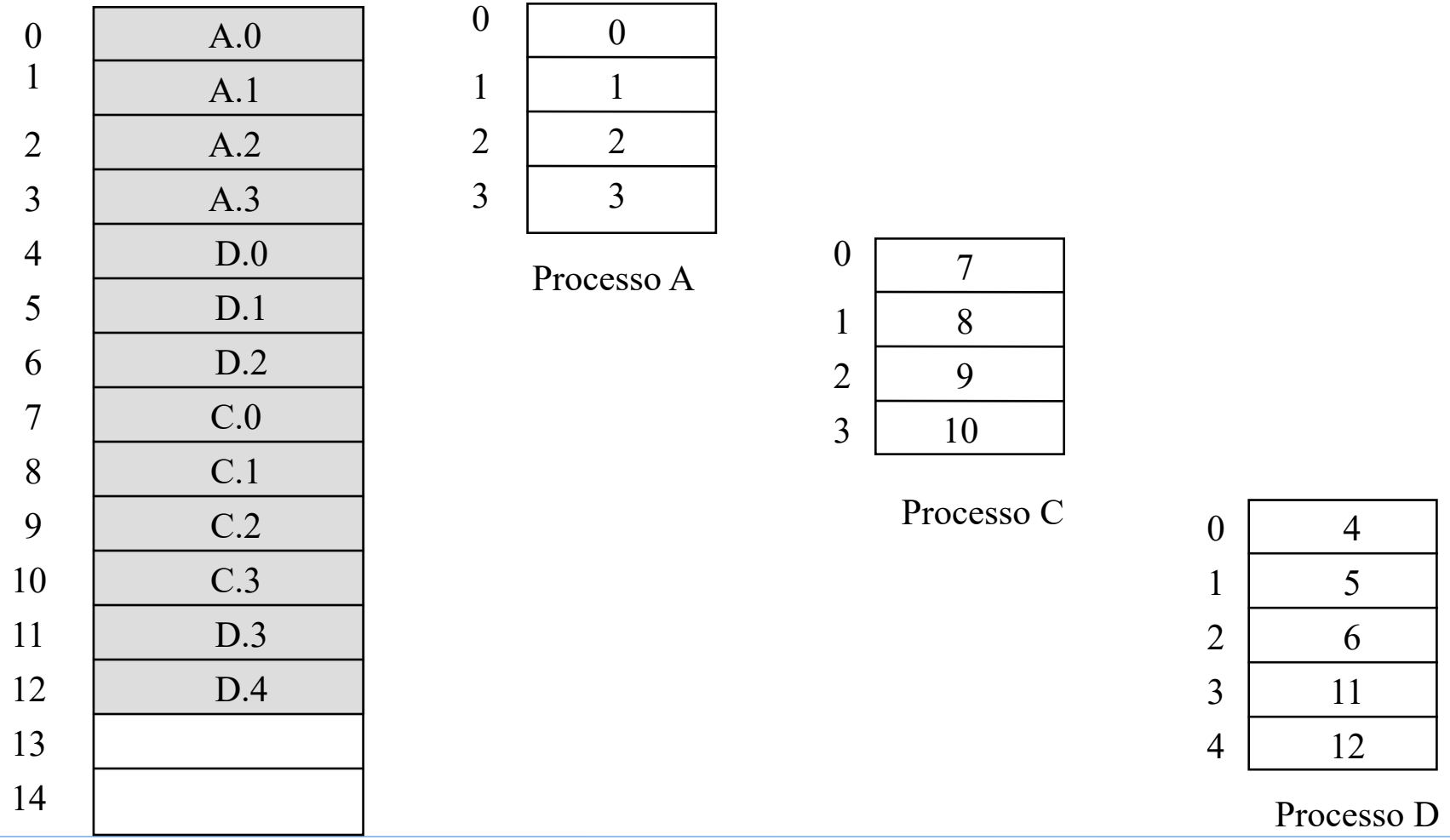
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

Gestão de memória física por páginas



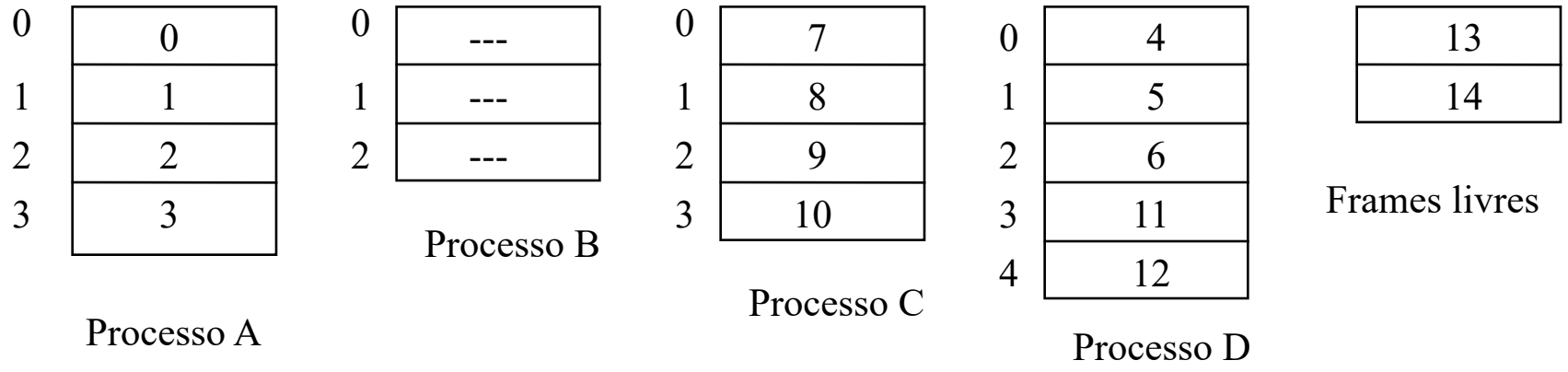
- **Transformação de endereços**: Hardware converte endereços virtuais em endereços físicos através de um tabela gerida pelo SO (tabela de páginas)

Paginação: uma tabela de páginas para cada processo



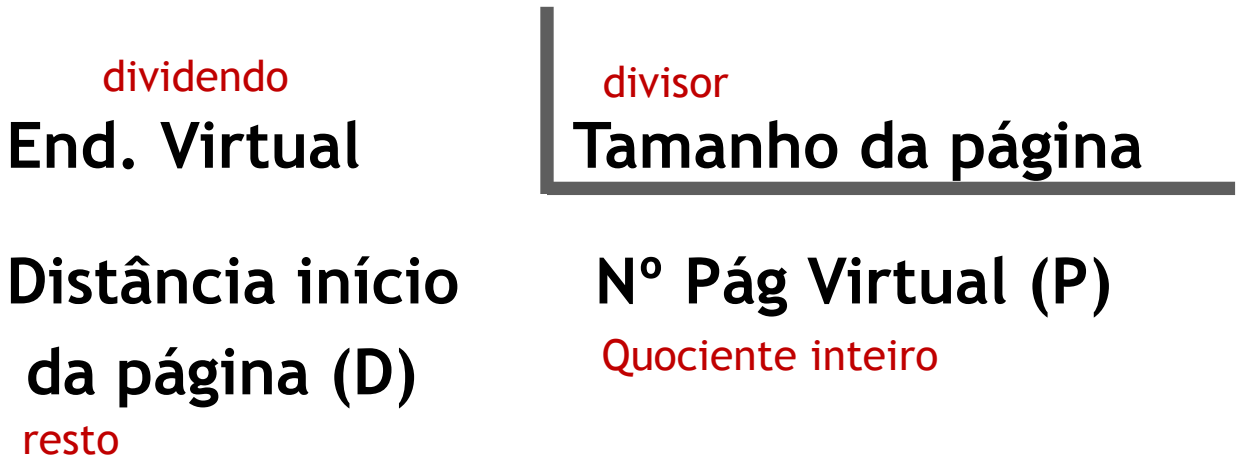
Tabelas de páginas para o exemplo

- O SO mantém uma tabela de páginas para cada processo:
  - Contém a “frame” correspondente a cada página do processo

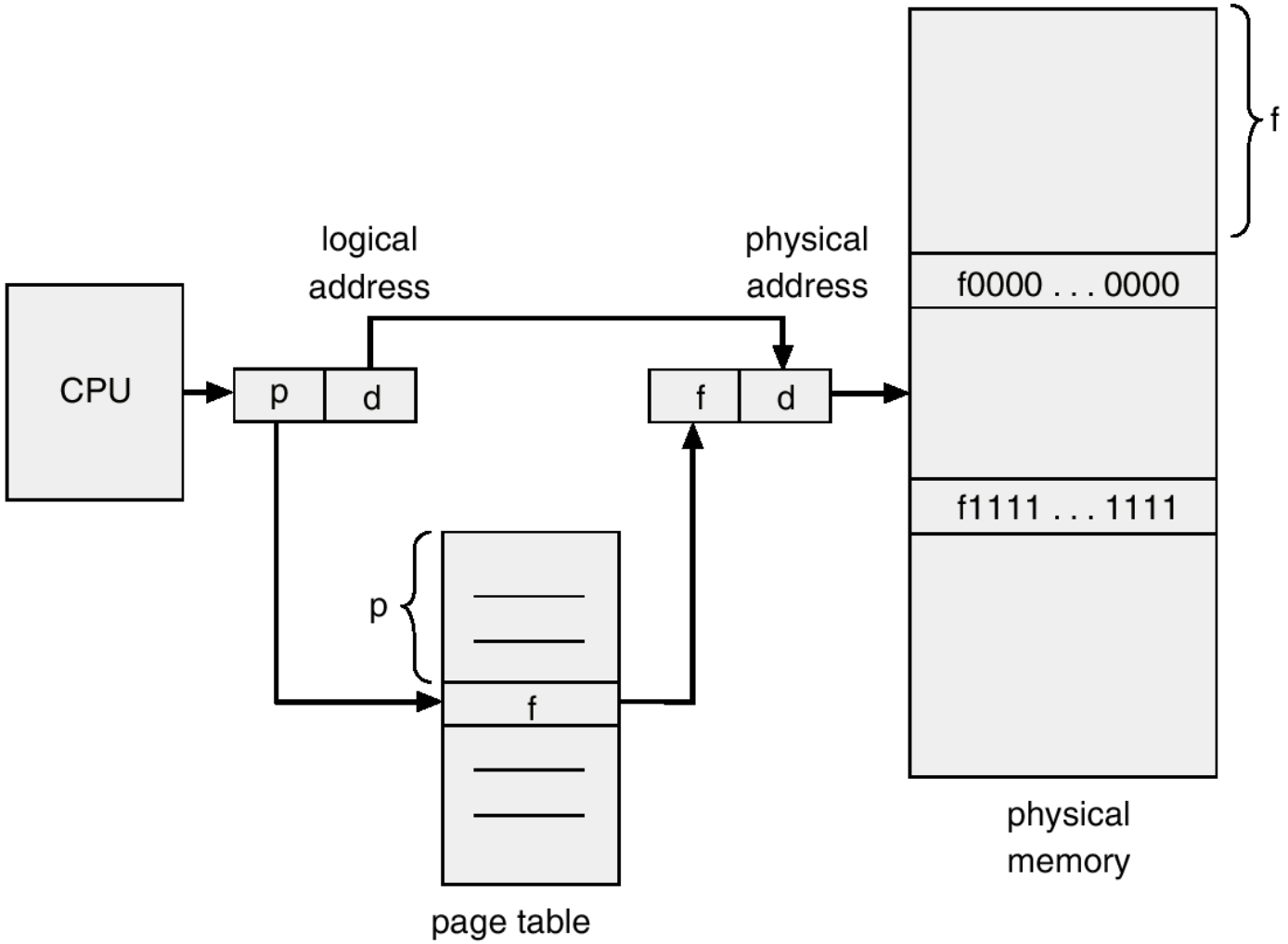


Transformação de endereços usada para paginação

- O endereço virtual (gerado pelo CPU) é composto por um número de página virtual (**P**) e um deslocamento (**D**)
- **P**: usado como índice na tabela de páginas que contém o endereço base de cada página na memória física.
- **D**: combinado com o endereço base define o endereço físico enviado para a memória

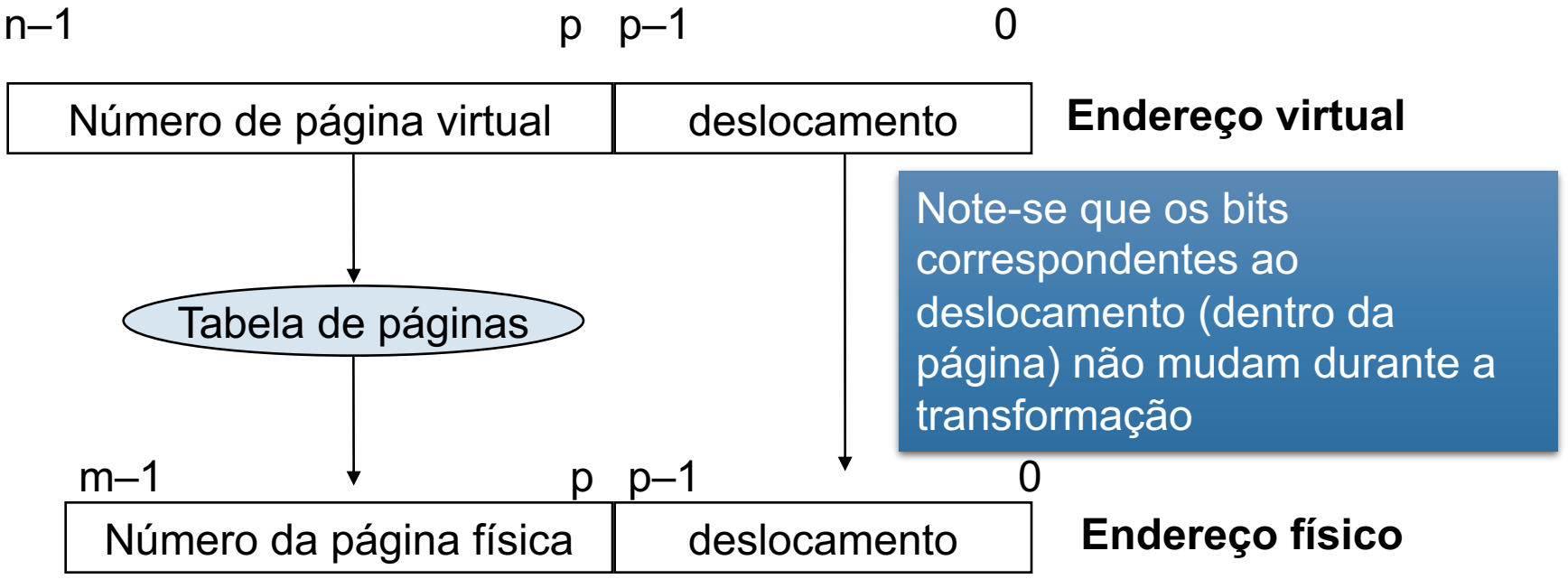


Transformação dos endereços virtuais em endereços físicos



Transformação de endereços virtuais em endereços físicos

- Parâmetros
  - P → 2^p = tamanho da página (bytes).
  - N → 2^n = Endereço virtual tem n bits
  - M → 2^m = Endereço físico tem m bits



Protecção de memória

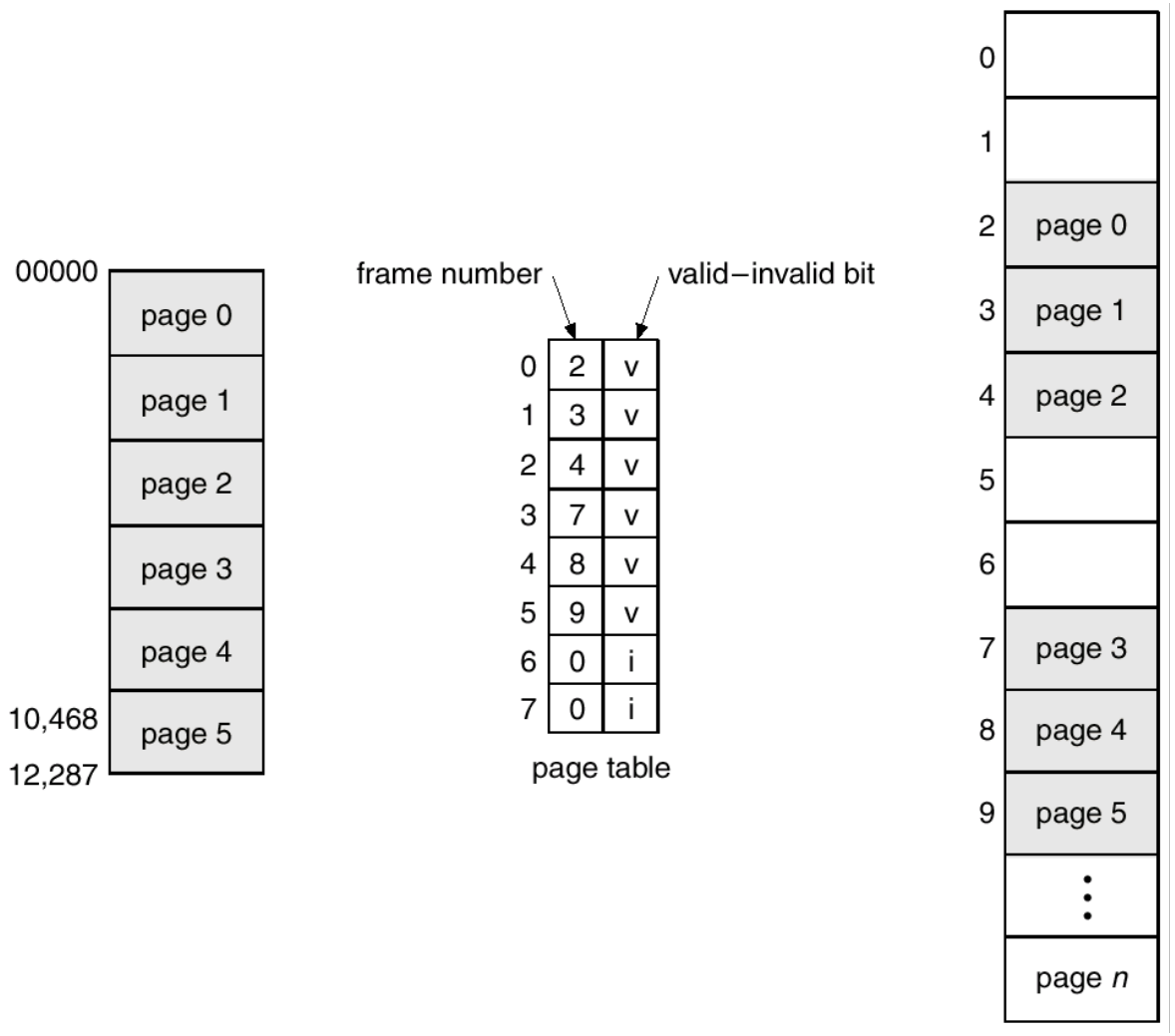
- A proteção de memória está associada ao preenchimento que o SO faz da tabela de páginas do processo

- Valid-invalid bit (bit V) associado a cada entrada da tabela de páginas:
  - "**válido**" indica que a página está no espaço de endereçamento lógico do processo – é uma página legal
  - "**inválido**" indica que a página não está no espaço de endereçamento virtual do processo.

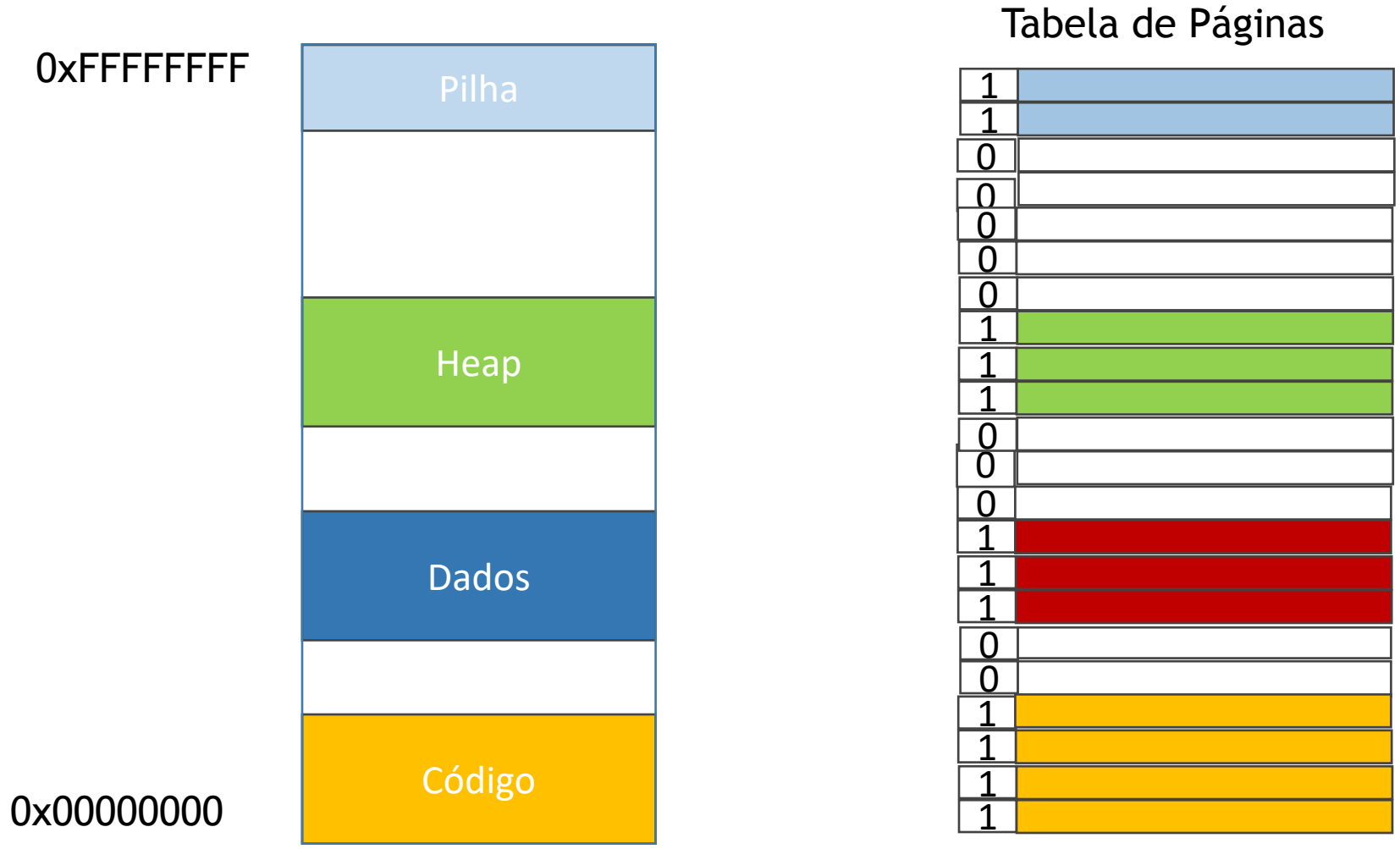
Informação para cada página

- A entrada I da tabela de páginas contém para a página I a seguinte informação:
  - Bit de validade V (página pertence ou não ao espaço de endereçamento do processo)
  - Número da frame F (se V==1)
  - Bits que definem as possibilidades de acesso: READ-ONLY, WRITE, EXECUTE
    - Páginas de código tem EXECUTE e READ-ONLY
    - Páginas de dados tem READ/WRITE
    - ...
  - Outros bits usados para gestão

Bit de validade na tabela de páginas



Bit de validade na tabela de páginas



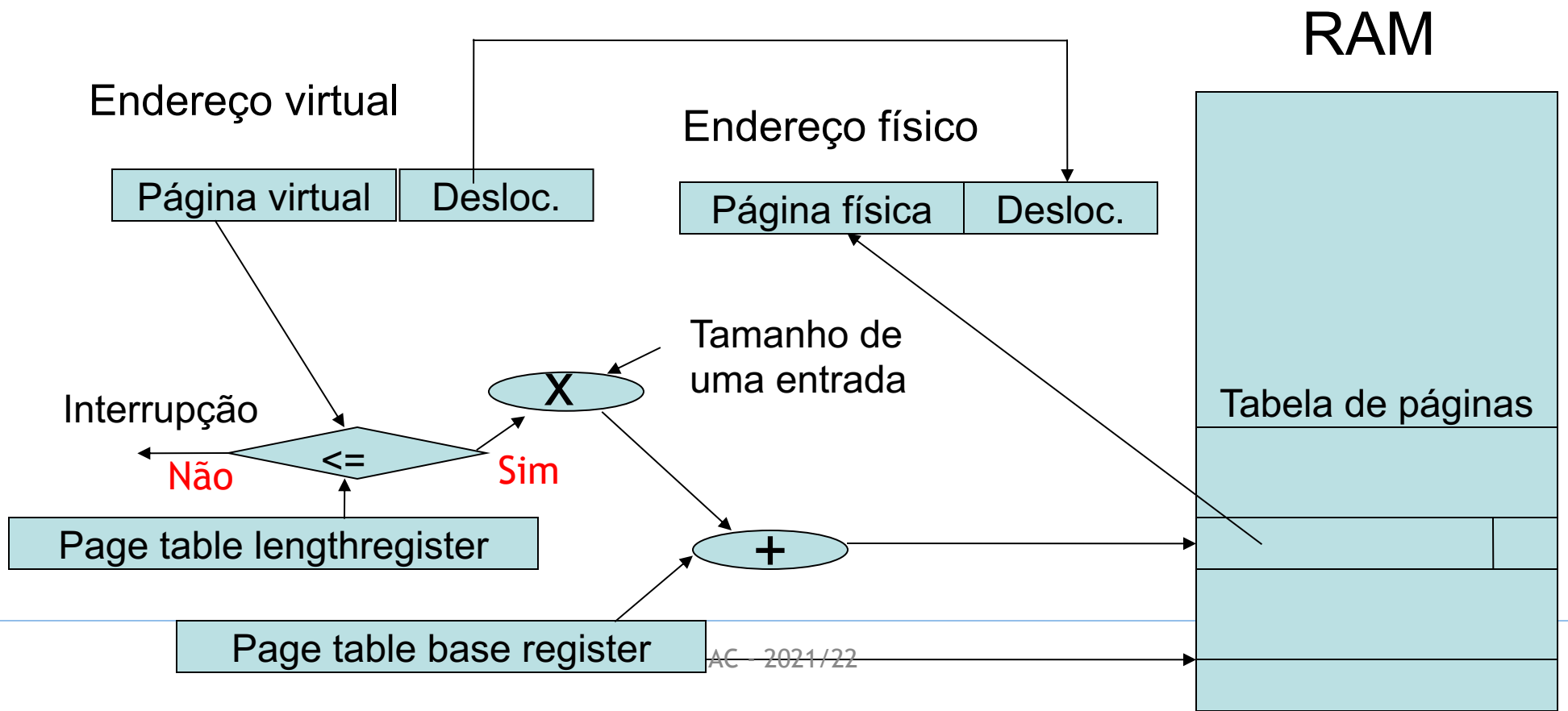
Suporte da tabela de páginas

- Onde se guarda a tabela de páginas de um processo?
  - No caso do Pentium a tabela tem 2^{20} entradas; se cada uma tiver 4 bytes, são 4 Mbytes …
  - Não é tecnicamente viável guardar a tabela de páginas na MMU
  - Há uma tabela de páginas por processo …

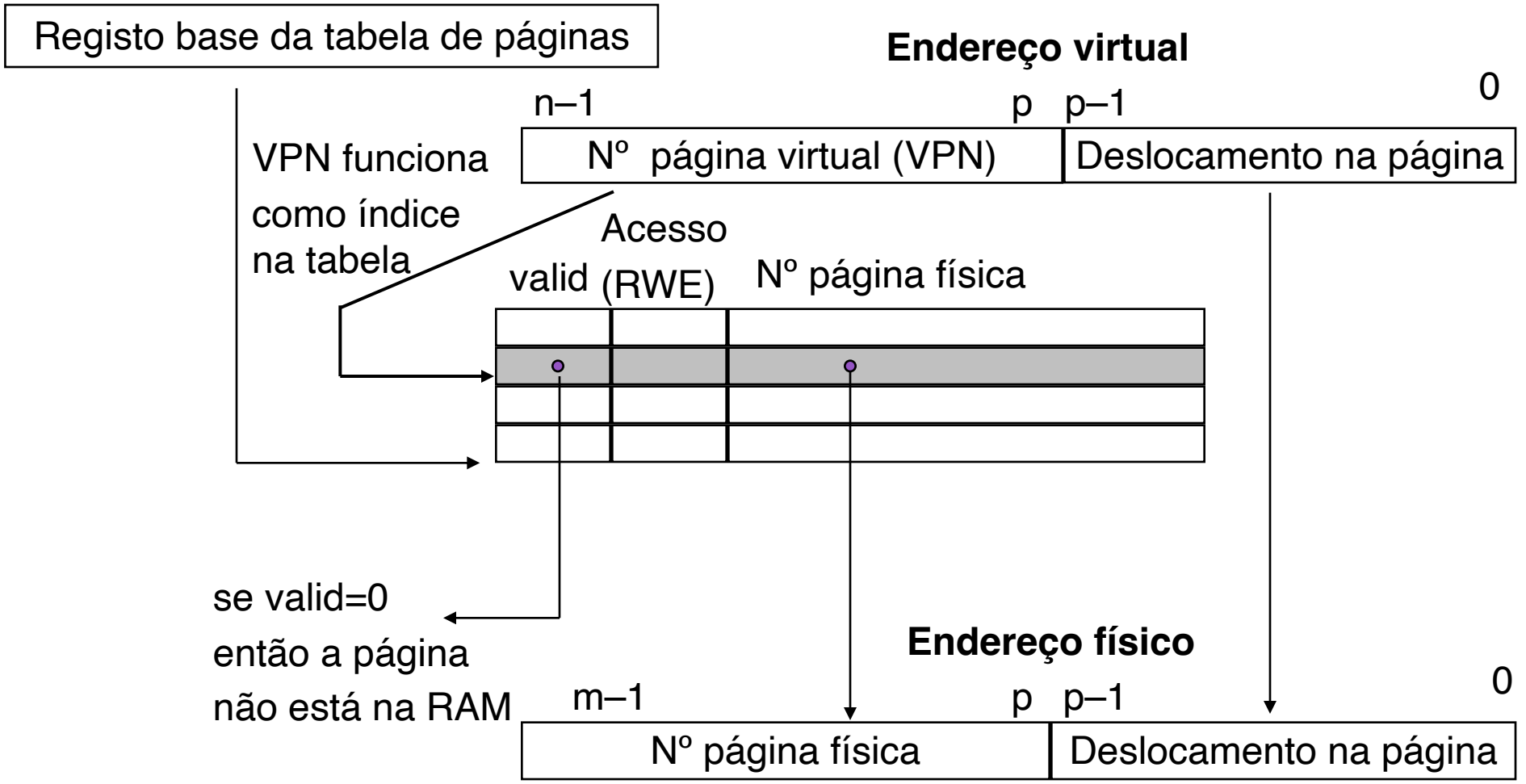
- **Solução**:
  - Guardar a tabela de páginas em memória

Suporte da tabela de páginas

- Tabela de páginas em memória central (RAM).
- *Page-table base register (*PTBR) aponta para a base da tabela.
- *Page-table length register* (PRLR) indica o tamanho da tabela de páginas.



Transformação de endereços usando a tabela de páginas



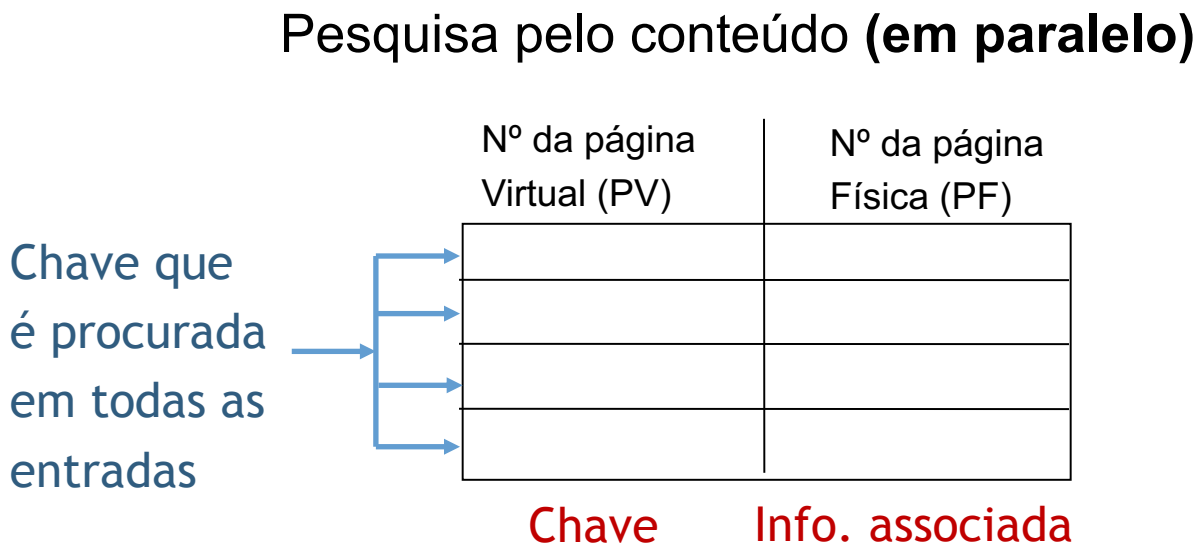
Suporte da tabela de páginas

- Neste esquema cada acesso a dados ou código **obriga a dois acessos à memória**: um para tabela de páginas e outra para o dado / instrução.

- O problema dos dois acessos à memória pode ser resolvido por hardware especial de consulta rápida chamado ***translation look-aside buffer (TLB)*** que se baseia numa *memória associativa* ou interrogável pelo conteúdo

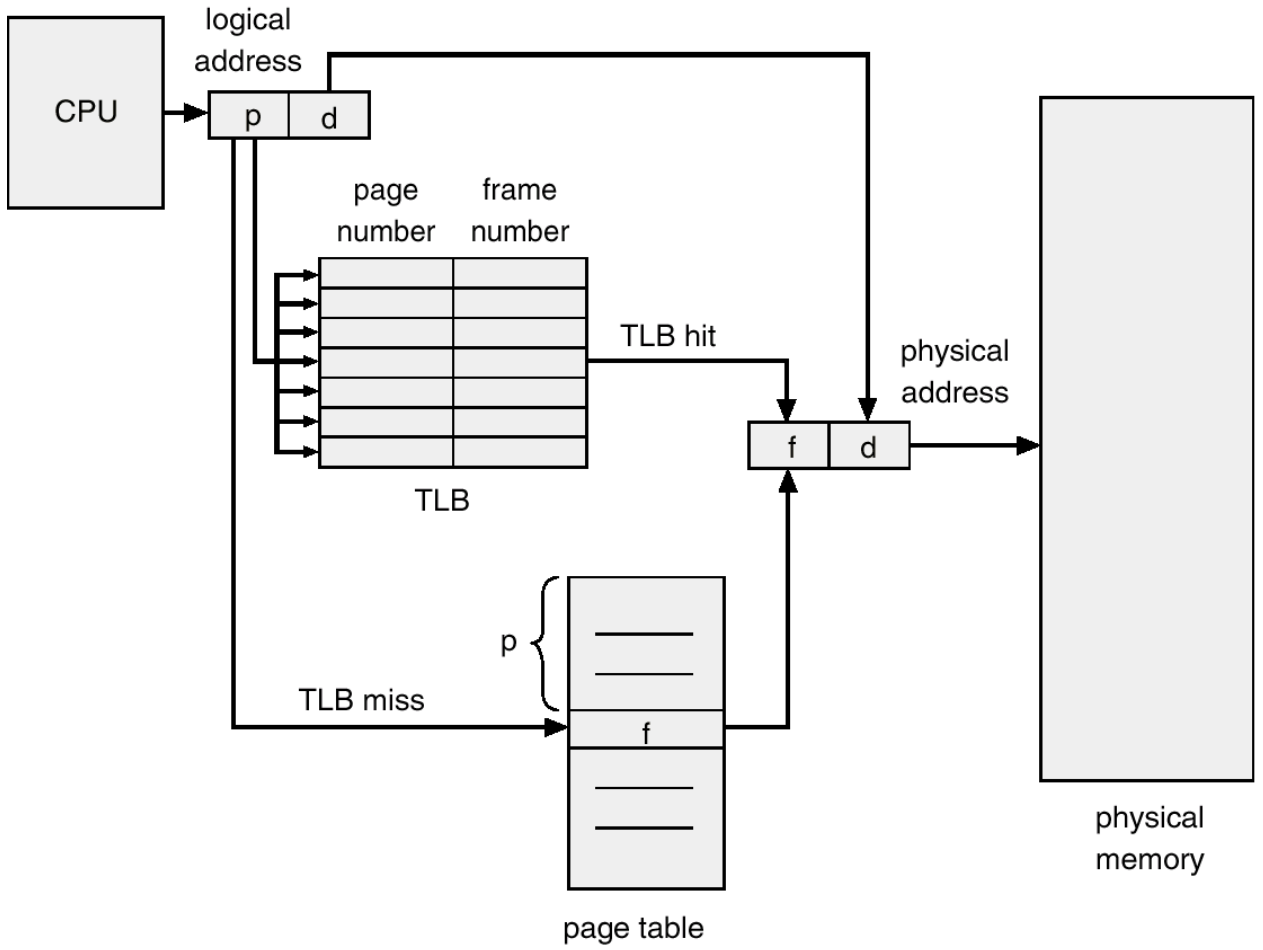
TLB - Memória associativa

- Transformação de endereços (PV, PF)
  - Se PV está na memória associativa, TLB responde com o número da "frame" PF.
  - Senão obter o número da "frame" PF da tabela de páginas na memória. Colocar o par (PV,PF) no TLB



Hardware de paginação com TLB

- "Translation Lookaside Buffer" (TLB)
  - pequena cache hardware na MMU
  - Faz a correspondência entre o nº página virtual em nº página física



Tamanho da página

- Quanto menor é o tamanho da página, menor é a memória desperdiçada por fragmentação interna
- Quanto menor o tamanho das páginas, mais páginas são precisas para um processo
- Mais páginas por processo significam tabelas de páginas maiores
- Maiores tabelas de páginas significa mais espaço ocupada em RAM pelas tabelas de páginas

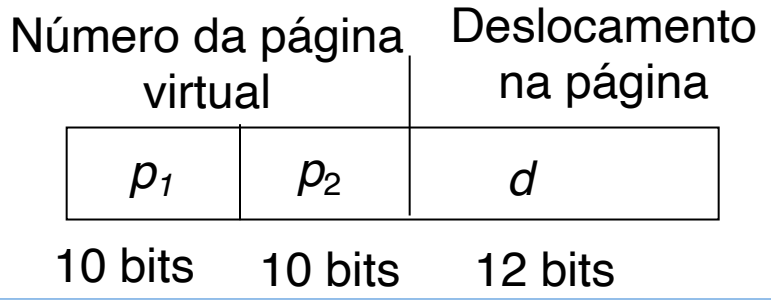
Tabela de páginas com mais de um nível

- O espaço de endereçamento virtual é dividido em várias partes, correspondendo a cada uma a sua tabela de páginas.

- Uma técnica simples é utilizar dois níveis (Pentium 32 bits).

Exemplo de tabela de páginas com 2 níveis: Pentium

- 32 bits de endereçamento, páginas de 4 Kbytes:
  - Número da página com 20 bits.
  - Deslocamento dentro da página (page offset) com 12 bits.
- A própria tabela de páginas ocupa várias páginas de 4 Kbytes, logo o número da página é dividido em:
  - a 10-bit para índice na tabela de páginas "principal" (***outer page table***)
  - a 10-bit para índice numa tabela de páginas "secundária" (***page of page table***)
- O endereço virtual é:

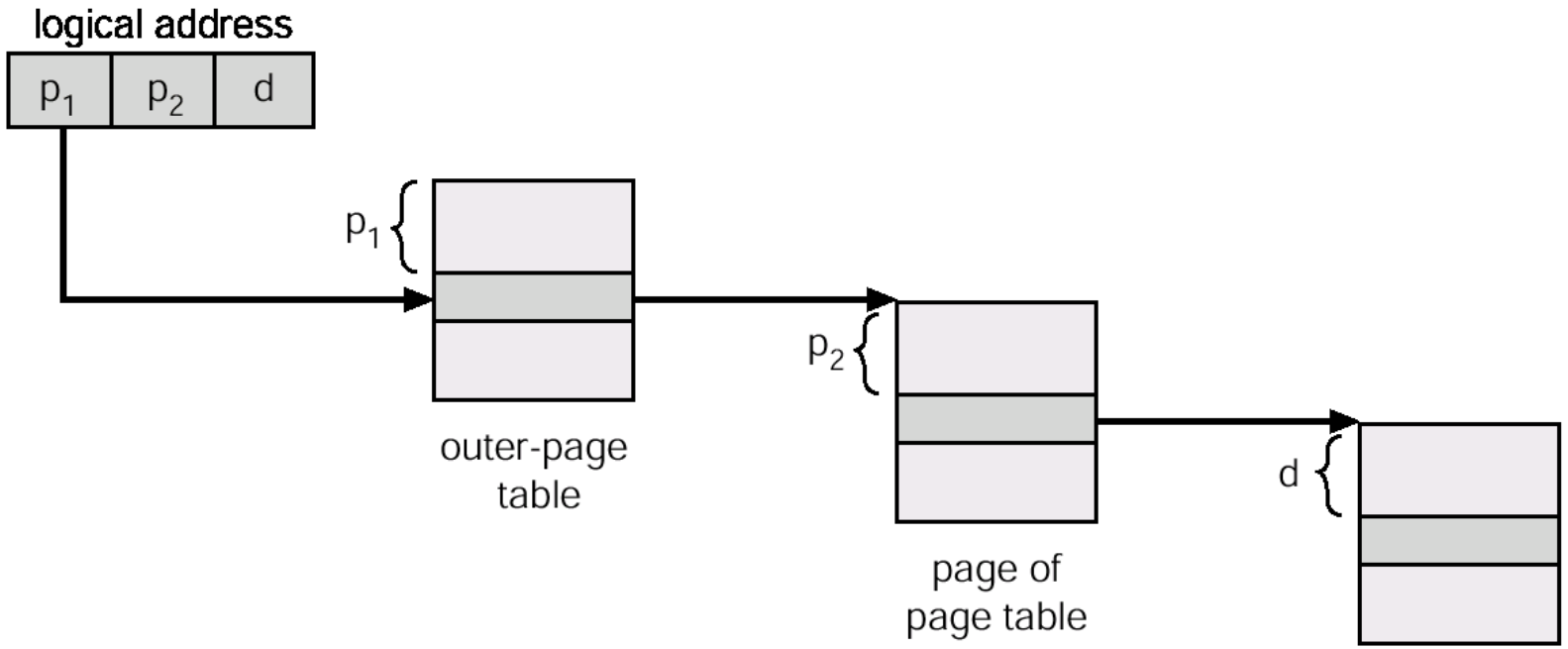


P1 → índice na tabela de páginas principal

P2 → índice na tabela de páginas secundária

Transformação de endereços: tabela de páginas com dois níveis

- Cada uma das tabelas de página vai ocupar uma página na memória física
- O TLB continua a conter pares (nº página virtual, nº da página física)



Tranformação de endereços: tabela de páginas com dois níveis

- Exemplo para o Pentium
  - 32 bits de endereço virtual, 20 bits para o nº da página virtual, 12 bits para o deslocamento
  - A “outer page table” chama-se page directory (PD) e as outras page tables (PT)
  - Se a tabela de páginas tivesse só um nível seria preciso ter sempre 2^{20} entradas na tabela de páginas. Se cada página contiver 1024 entradas da tabela de páginas são precisas 1024 páginas
- Se a tabela de páginas tiver 2 níveis
  - A page directory existe sempre (1 página)
  - Se o espaço de endereçamento do processo for pequeno, só P entradas da PD estão preenchidas com P << 1024 (P páginas)

Motivações para a memória virtual

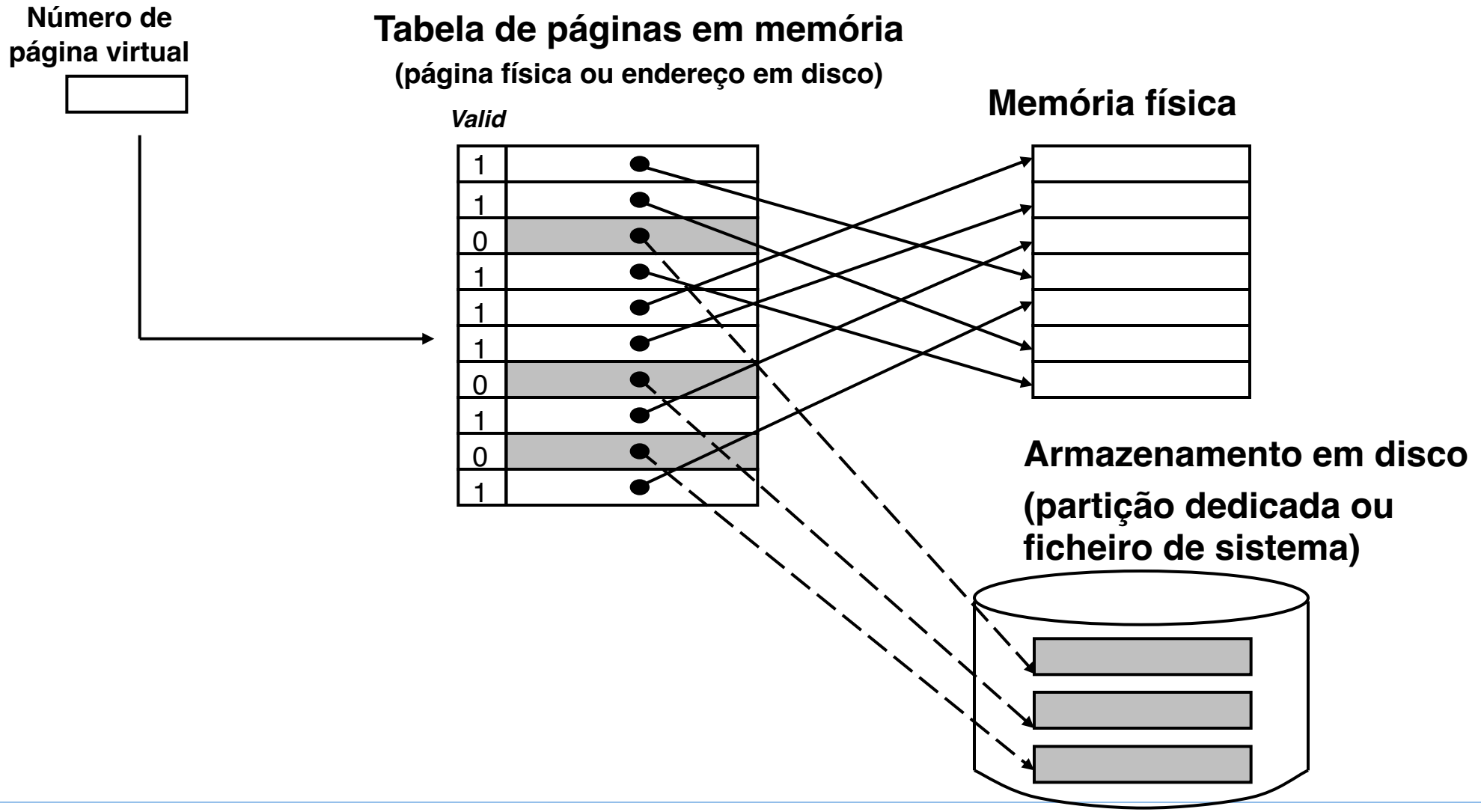
- Usar a RAM como **Cache para o Disco**
  - O espaço de endereçamento do processo pode exceder a dimensão da memória.
  - A soma dos espaços de endereçamento dos vários processos **pode exceder** o tamanho da memória física
- Simplificar a gestão de memória
  - Múltiplos processos residem em memória central cada um com o seu próprio espaço de endereçamento.
    - Só o código e dados “ativos” é que **estão** em RAM
    - mais RAM pode ser atribuída se necessário.
  - Fornece **proteção** – isto já era conseguido a partir da tabela de páginas mesmo sem memória virtual.

Paginação a pedido

- A página só é trazida para memória quando é referenciada.
  - Menos I/O
  - Menos RAM utilizada
  - Tempo de resposta menor
  - Mais programas em RAM

- Página é necessária $\Rightarrow$ referencia-se
  - Se não está em memória gera-se uma interrupção.
  - O SO verifica se se trata de:
    - Referência inválida $\Rightarrow$ abortar a execução
    - Não-em-memória $\Rightarrow$ é preciso trazer a página virtual para RAM

Tabela de páginas



Transformação de endereços

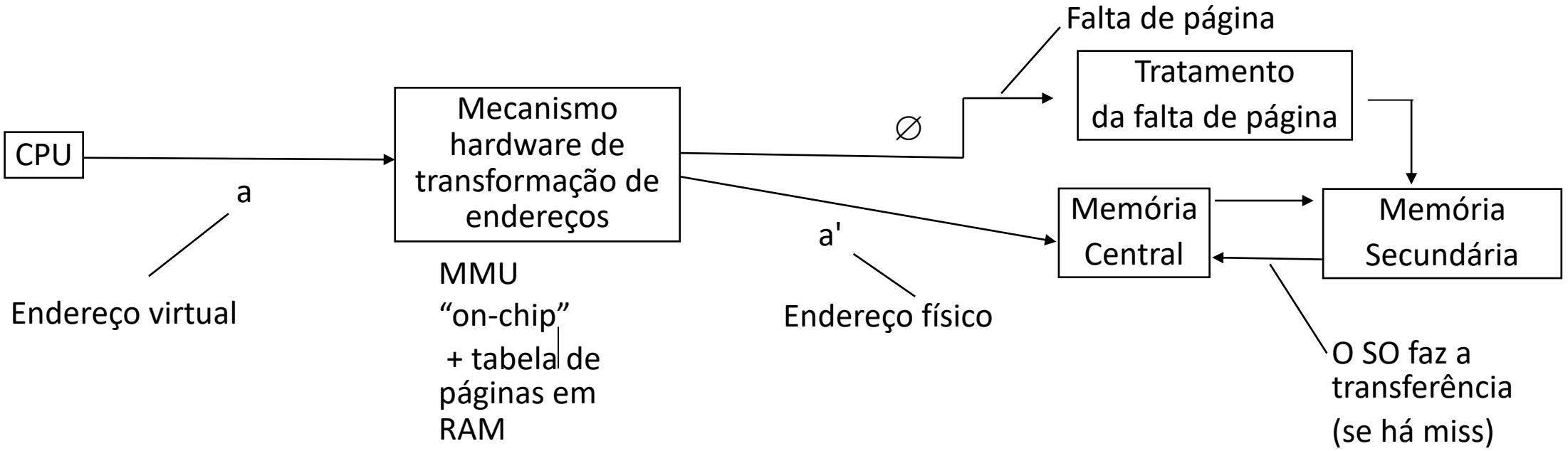
V = {0, 1, . . . , N–1} espaço de endereçamento virtual

P = {0, 1, . . . , M–1} espaço de endereçamento físico **N > M**

MAP: V → P U {∅} função de correspondência

MAP(a) = **a'** se o dado no endereço **a** está presente no endereço físico **a'** em P

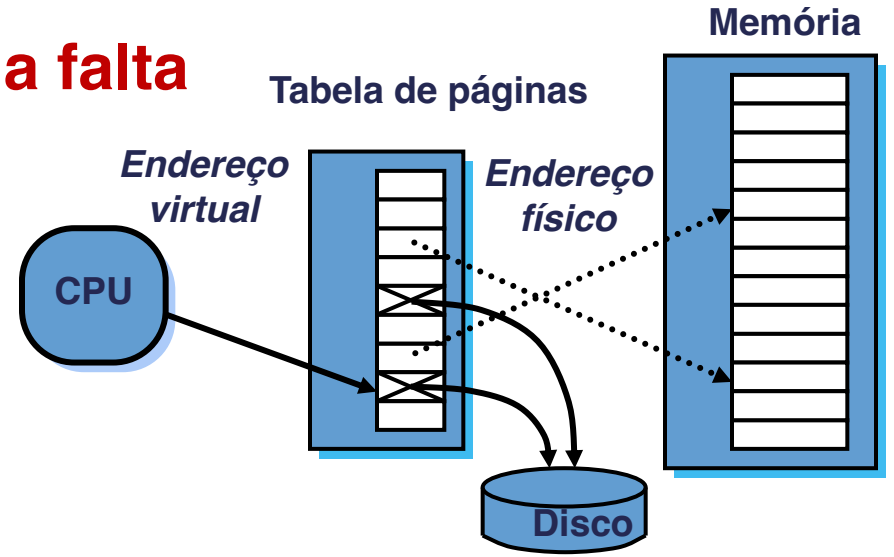
= ∅ se o dado no endereço virtual **a** não está presente em P



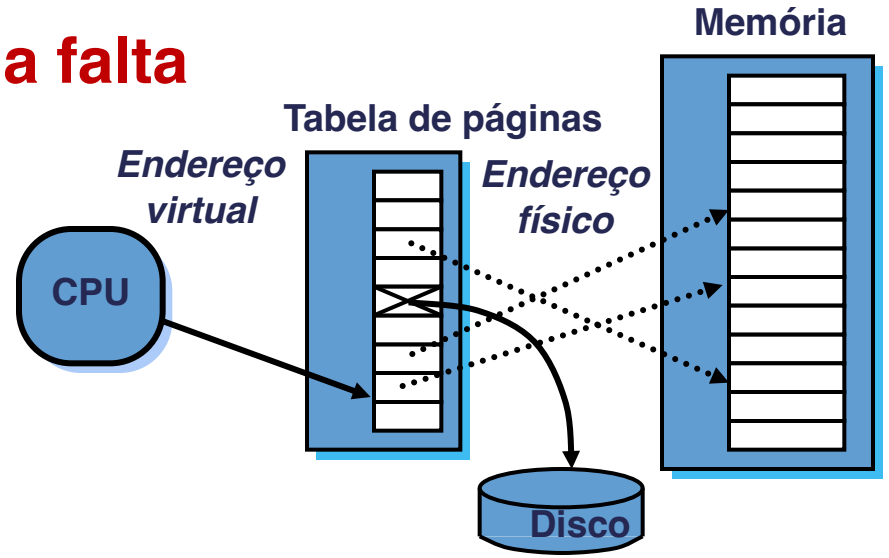
Falta de página

- Tabela de páginas indica que o conteúdo referenciado pelo endereço virtual não está em RAM
- É invocado um procedimento de exceção para trazer os dados para memória
  - Enquanto a página em falta é carregada, o processo que fez a referência de memória pode ceder o CPU a outros processos

Antes da falta

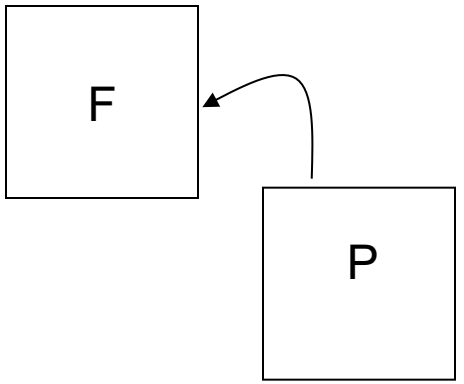


Depois da falta



Rotina de tratamento da Falta de Página

1. A primeira referência a uma página P por um processo X provoca uma exceção (fault) e o SO intervém
2. SO olha para a tabela de páginas de X para decidir se é:
   - Referência inválida ⇒ aborta o programa.
   - Falta de página.
3. Se for uma falta de página, obtém uma página física não utilizada (F).
   - Se existe uma frame F livre, seleciona-a
   - Caso contrário
     I. O SO faz uso de um **algoritmo de substituição de páginas (a estudar em FSO)** que vai escolher F
     II. Se F está "**suja**" é preciso escrever o seu conteúdo no disco
     III. Modifica a tabela de páginas do processo (Y) a quem pertence a página em F.
     Se Q é a página guardada em F: tab_paginas_de_Y[Q].V = 0



Rotina de tratamento da Falta de Página

4. Programa o controlador do disco para transferir o bloco em que está a página P do processo X para a frame F.
   - Enquanto espera que a transferência ocorra, o processo X não pode prosseguir. O SO atribui o CPU a outro processo.

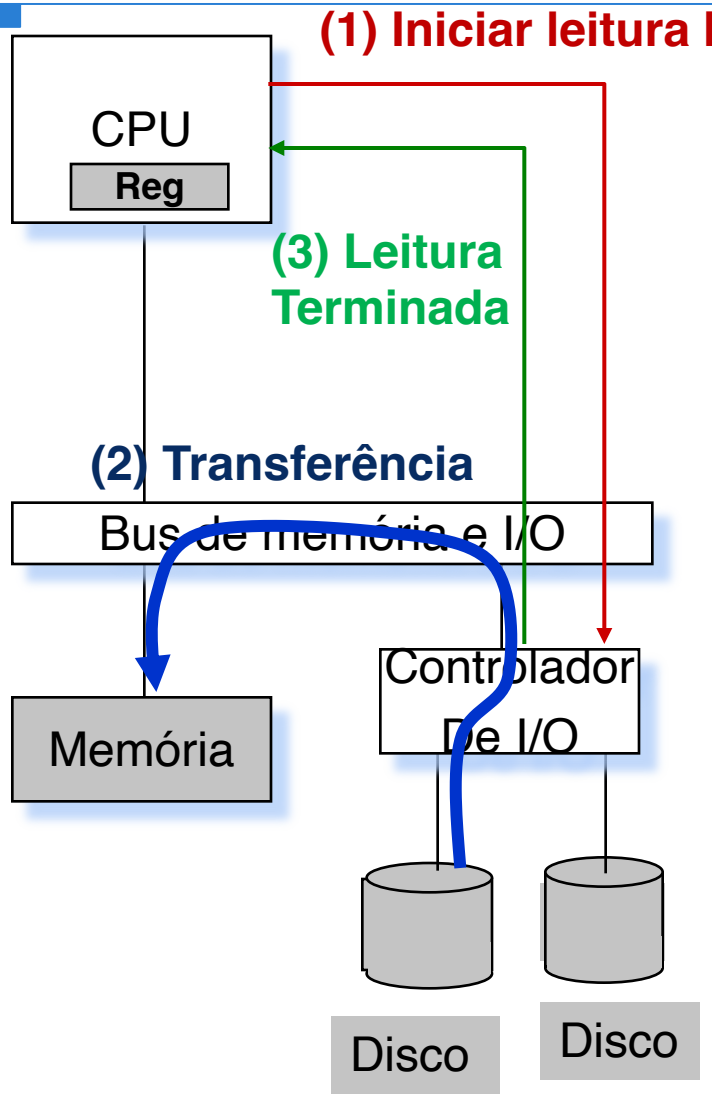
5. Quando a transferência termina, há uma nova interrupção e a rotina de tratamento correspondente modifica a tabela de páginas do processo X

```
tab_paginas[P].frame = F
tab_paginas[P].V = 1
```

- O processo X pode retomar a sua execução na instrução que provocou a exceção

Atendimento da falta de página

1. CPU pede ao periférico para ler um bloco de dimensão P a partir do endereço X e escrever na RAM começando no endereço Y
2. Ocorre a leitura
   - O periférico transfere os dados para a RAM
3. Periférico assinala o fim da transferência com uma interrupção
   - SO atualiza a tabela de páginas
   - Processo pode continuar



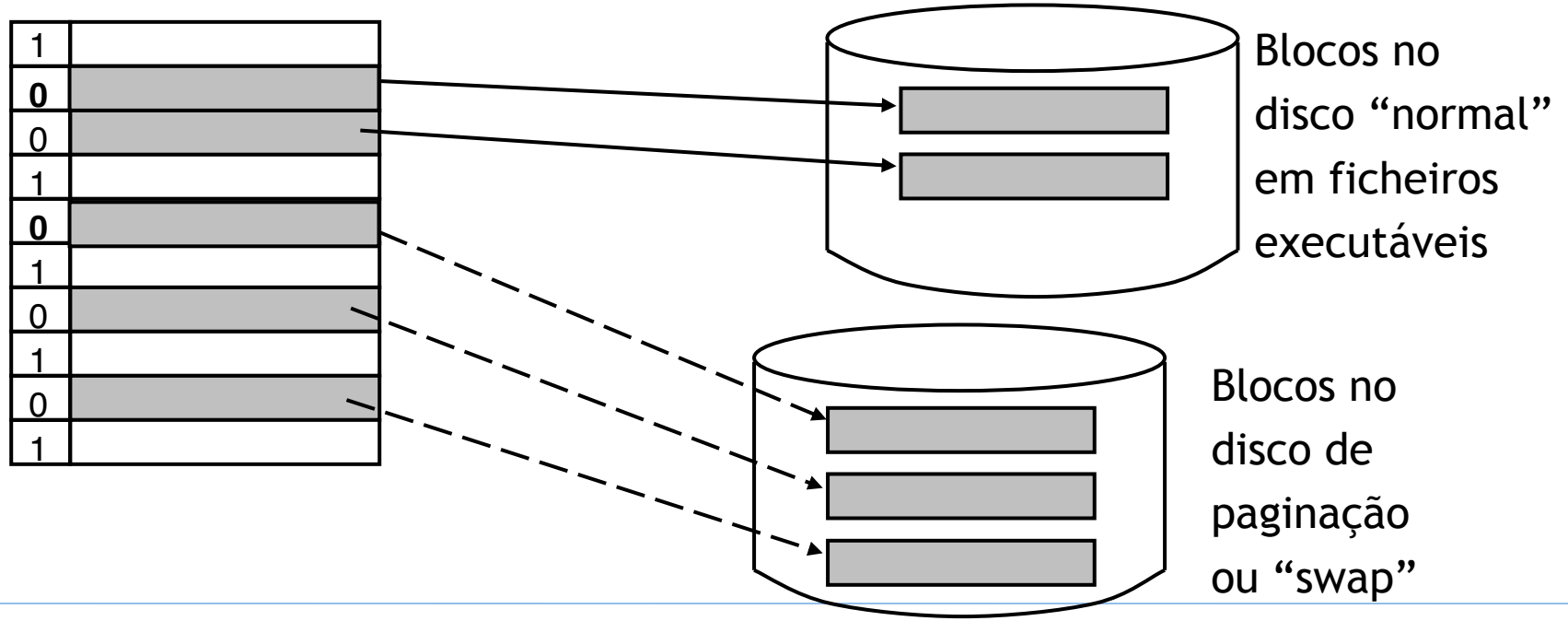
Onde estão as páginas no disco?

- Páginas de Código (read-only)
  - No ficheiro executável

- Páginas que podem ser alteradas (dados, heap, pilha)
  - No disco de paginação ou swap
  - Páginas podem estar em dois estados
    - Limpas (não foram alteradas desde que foram carregadas)
      - Podem ser reutilizadas imediatamente, porque há uma cópia no disco de swap
    - Sujas (foram alteradas desde que foram carregadas)
      - Antes de serem utilizadas têm de ser escritas no disco de swap

Onde estão as páginas no disco?

- Páginas de Código (read-only)
  - No ficheiro executável
- Páginas que podem ser alteradas (dados, heap, pilha)
  - No disco de paginação ou swap



- Há muito mais para aprender sobre a gestão de memória virtual, em particular sobre paginação a pedido.
  - Algoritmo de substituição
  - Que frames são candidatas a vitimas (próprio processo ou outros)
  - Como recuperar rapidamente de uma má escolha de vitima

- Tudo isso será abordado em FSO