Laboratório de Pesquisa em Redes e Multimídia

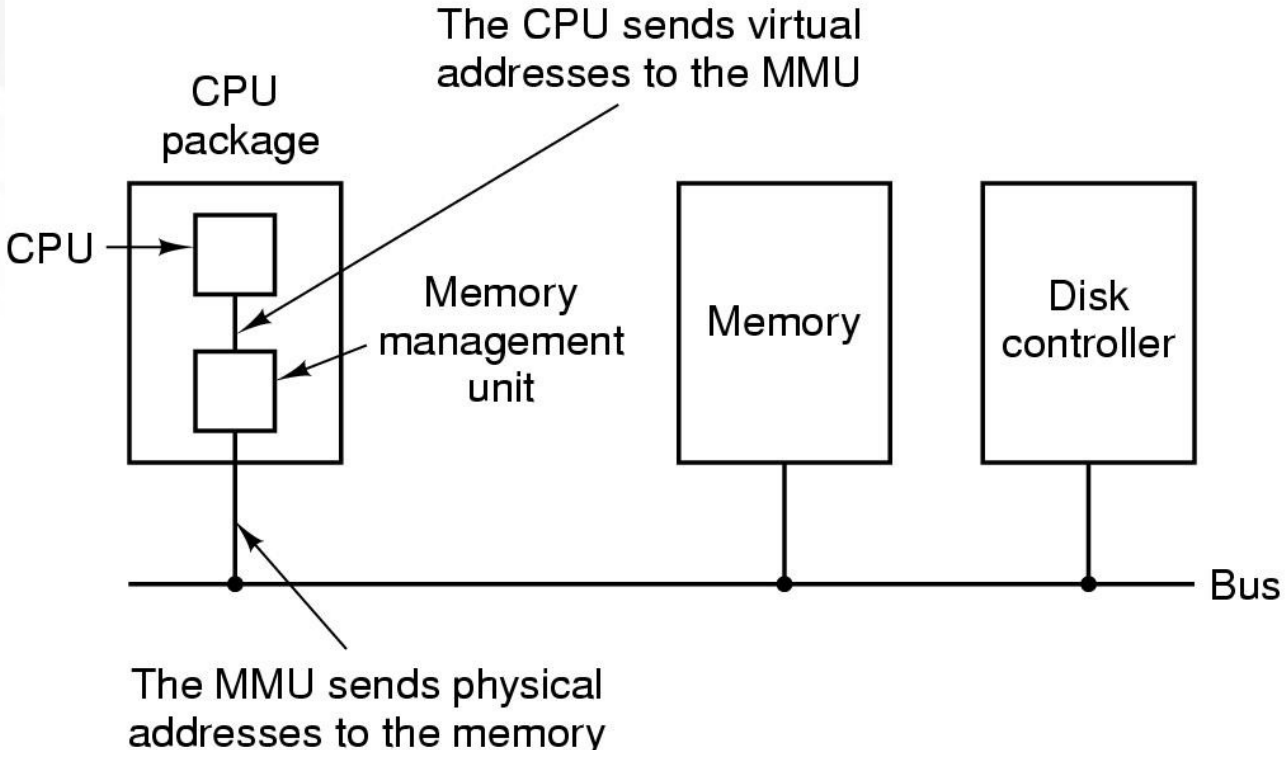
Gerência de Memória

Paginação

Universidade Federal do Espírito Santo
Departamento de Informática

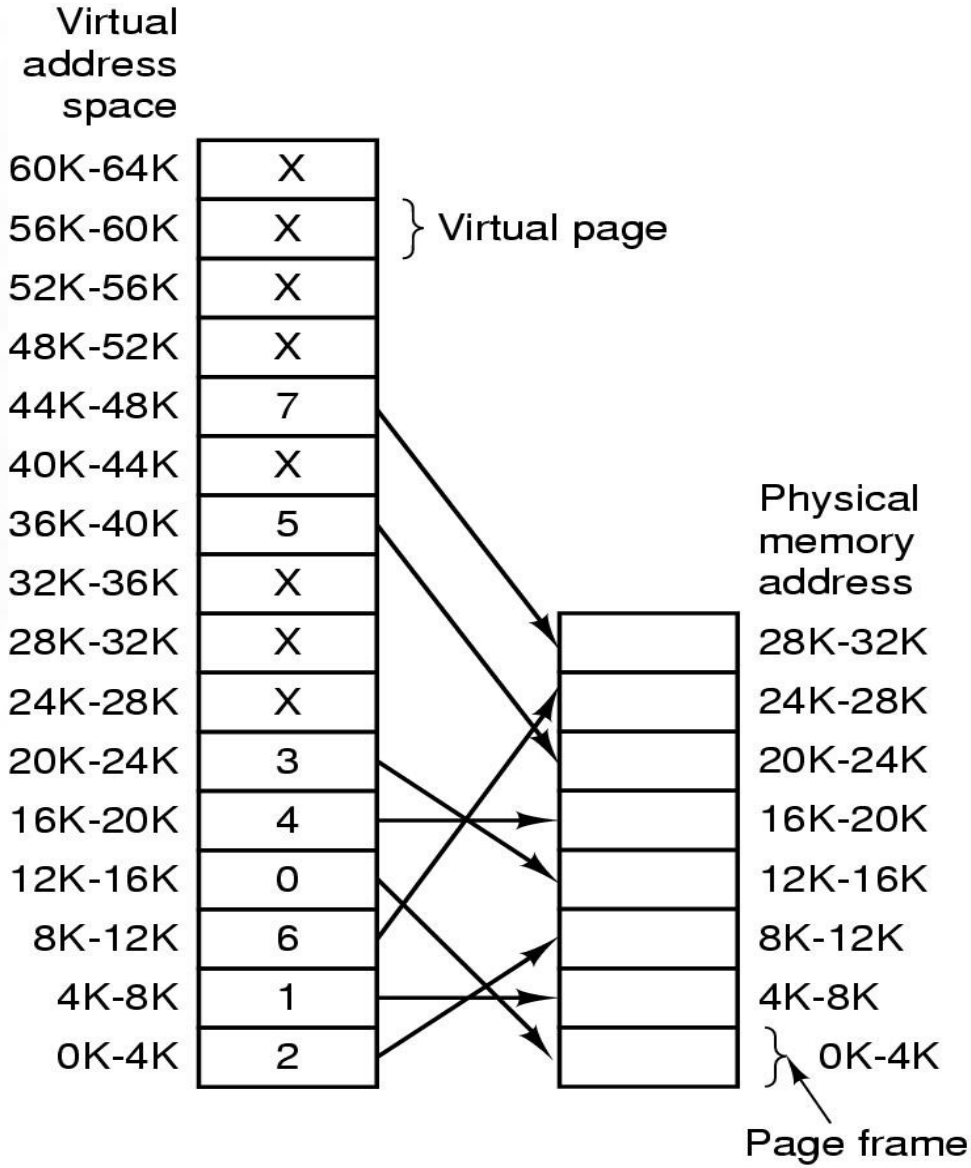
Endereçamento Virtual (1)

- O programa usa endereços virtuais
- É necessário HW para traduzir cada endereço virtual em endereço físico
  - MMU: Memory Management Unit
  - Normalmente no mesmo chip da CPU

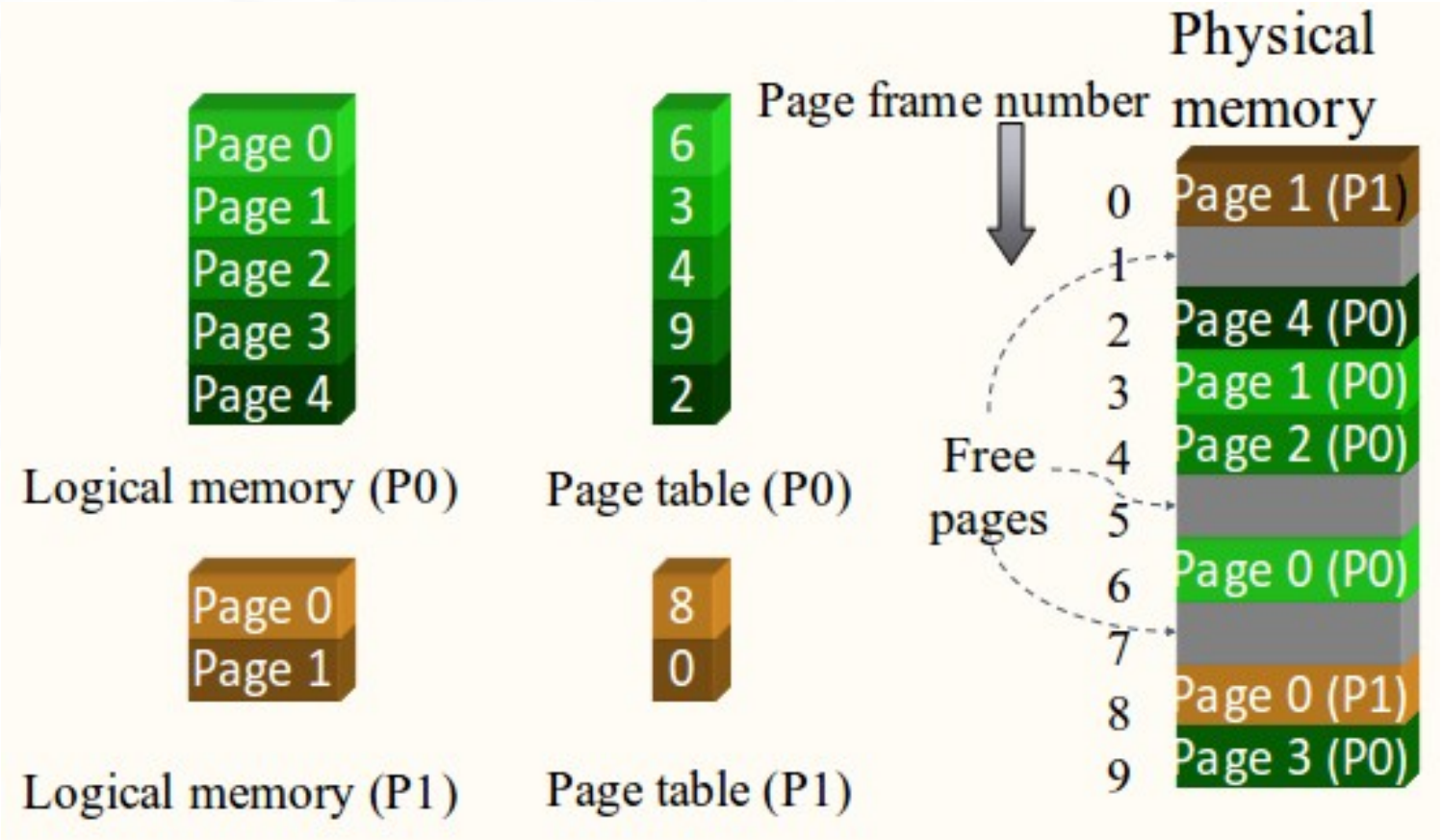


Endereçamento Virtual (2)

- Exemplo
  - Computador capaz de gerar endereços virtuais de 16 bits (0->64k).
  - Memória física de apenas 32k => programas não podem ser carregados por completo na memória física
  - Solução: dividir o pro-grama em **Páginas**

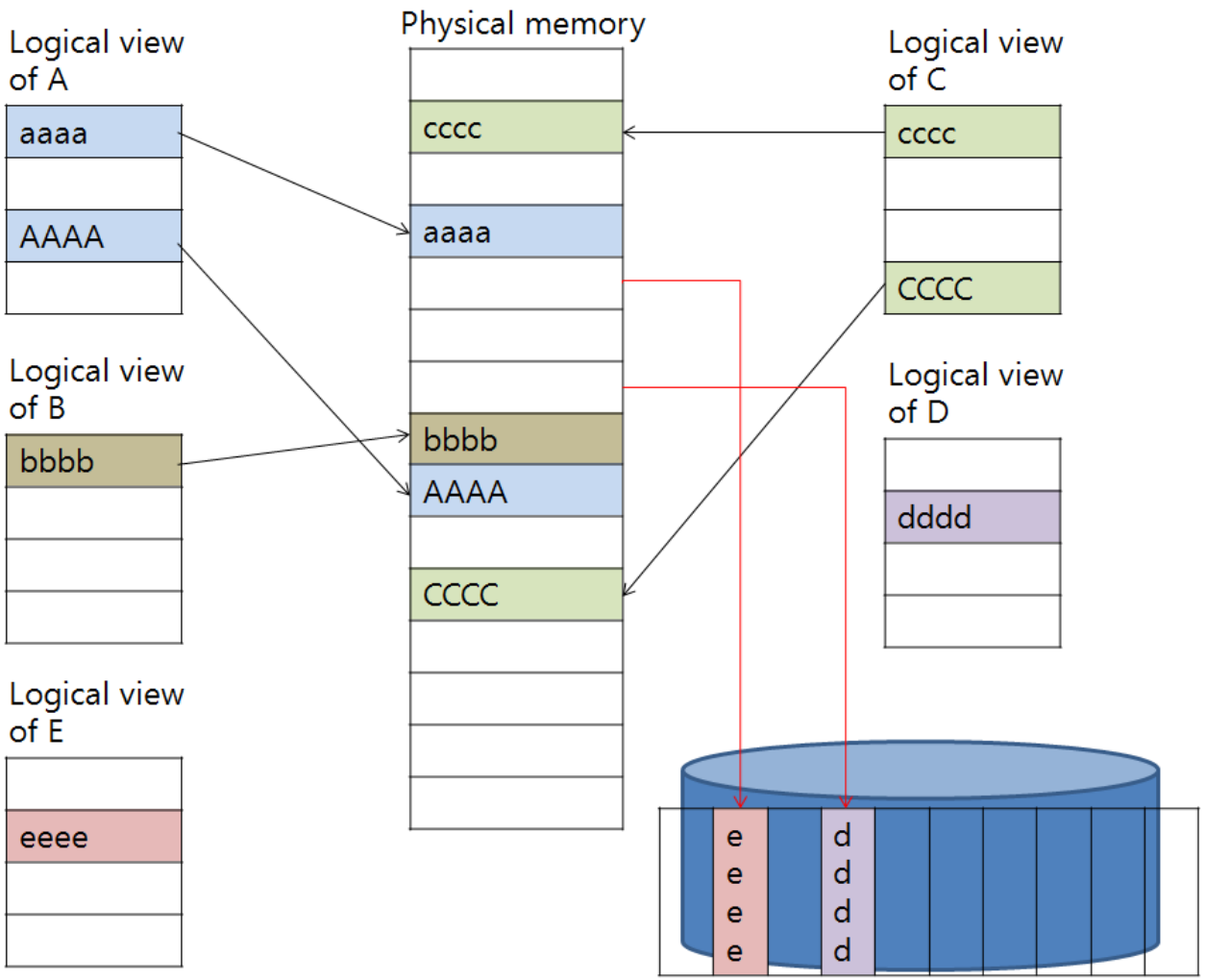


Endereçamento Virtual (3)



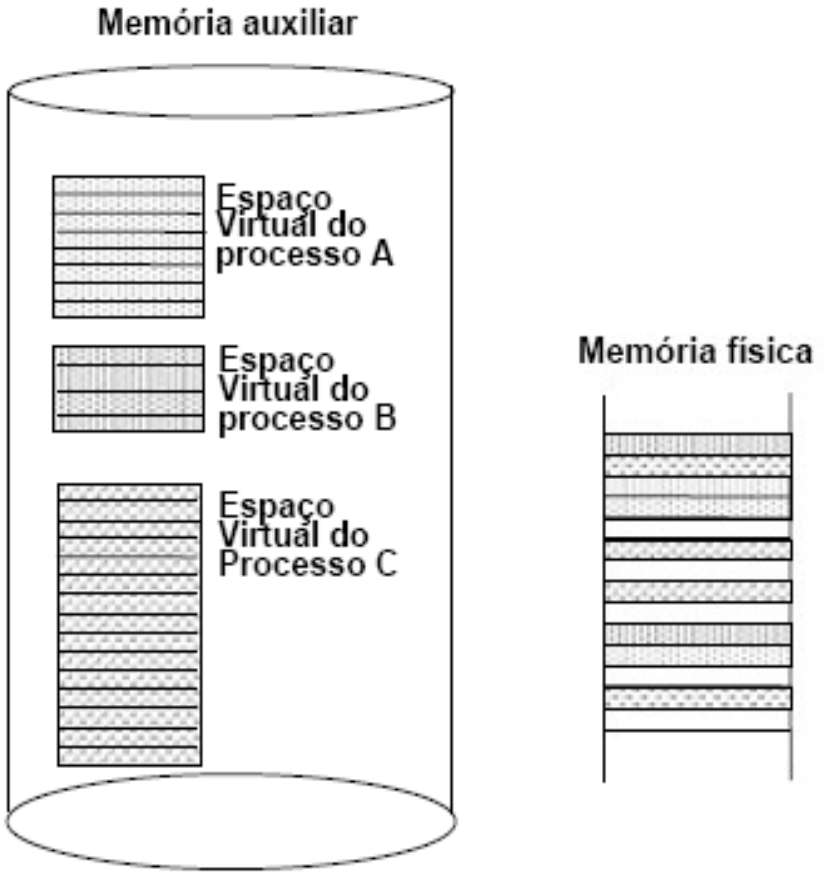
Memória Secundária: Área de Swap

Apenas precisam estar na memória principal as páginas que estão sendo utilizadas por cada processo!



Memória Secundária: Área de Swap

- Inicialmente...
  - Exemplo: uma cópia completa do programa deve estar presente em disco, de modo que partes (páginas) possam ser carregadas dinamicamente na memória quando necessário
-



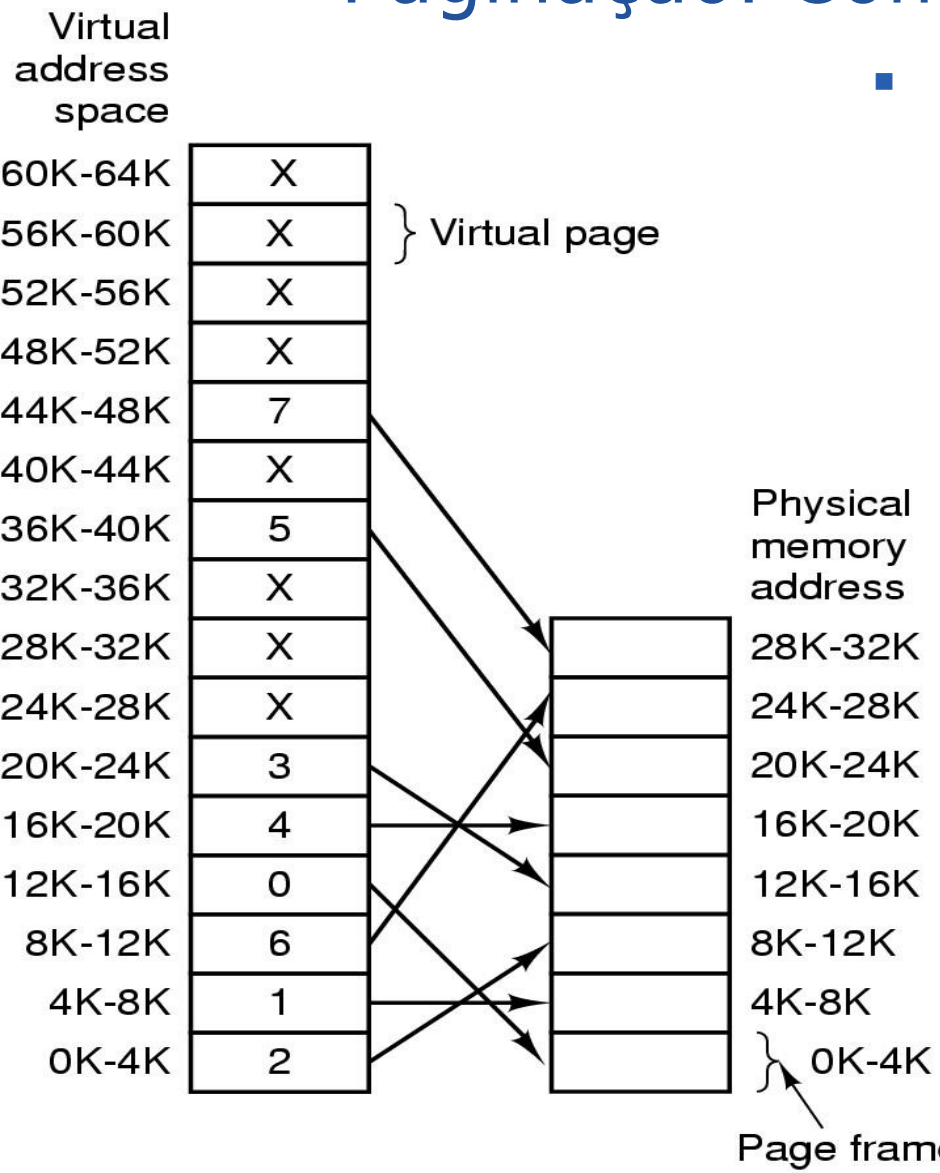
Memória virtual: Paginação

- Processo é dividido em **Páginas**
- A Memória é dividida em **Molduras** (ou *Frames*) de mesmo tamanho
  - Tamanho das Páginas = tamanho das Molduras
- Páginas/Molduras são de pequeno tamanho (e.g., 1K):
  - fragmentação interna pequena
- Processo não precisa ocupar área contígua em memória
  - Elimina fragmentação externa
- Processo não precisa estar completamente na MP
- SO mantém uma **tabela de páginas** por processo
- Endereços são gerados dinamicamente em tempo de execução

Paginação: Como funciona?

- Para minimizar a informação necessária à conversão, a memória virtual é logicamente dividida em páginas
  - Endereço virtual = (nº da página , deslocamento)
- No exemplo anterior (end. virtuais de 16 bits=> processos de até 64k; memória física de 32k; páginas/molduras de 4k)
  - São necessários 4 bits para referenciar todas as 16 páginas do processo
    - End. virtual = (nº da página [4 bits] , deslocamento [16 – 4 bits])
  - Instrução MOV REG, 0
    - O end. virtual 0 é enviado à MMU
    - Ela detecta que esse end. virtual situa-se na página virtual 0 (de 0 a 4095) que, de acordo com o seu mapeamento, corresponde à moldura de página 2 (end. físicos de 8192 – 12287)

Paginação: Como funciona? (2)



- `MOV REG, 20500`
  - **Qual é a página?**

    Pag. 5, que contém os endereços de 20k (20480) até 24k-1 (24575)
  - **Esta página está em qual moldura?**

    Na moldura 3, que contém end. físicos de 12k (12288) a 16k-1 (16384)
  - **Qual o deslocamento do endereço 20500 dentro da página?**

    Desl. = End. virtual – End. virtual do 1º byte da página

    = 20500 – 20480= 20
  - **Qual será o endereço físico correspondendo ao end. virt. 20500?**

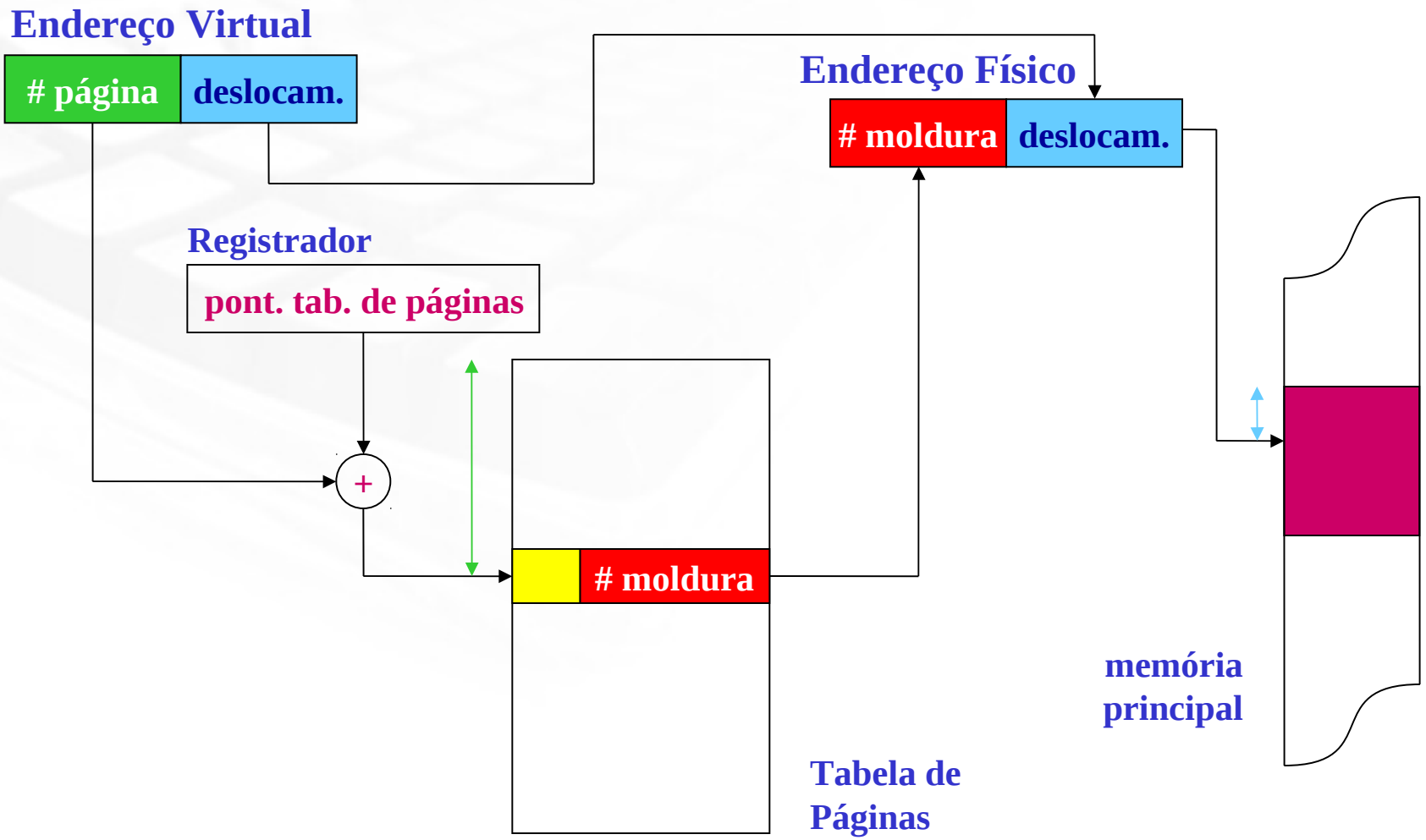
    = End. do 1º byte da moldura + desloca.

    = 12288 + 20 = 12308

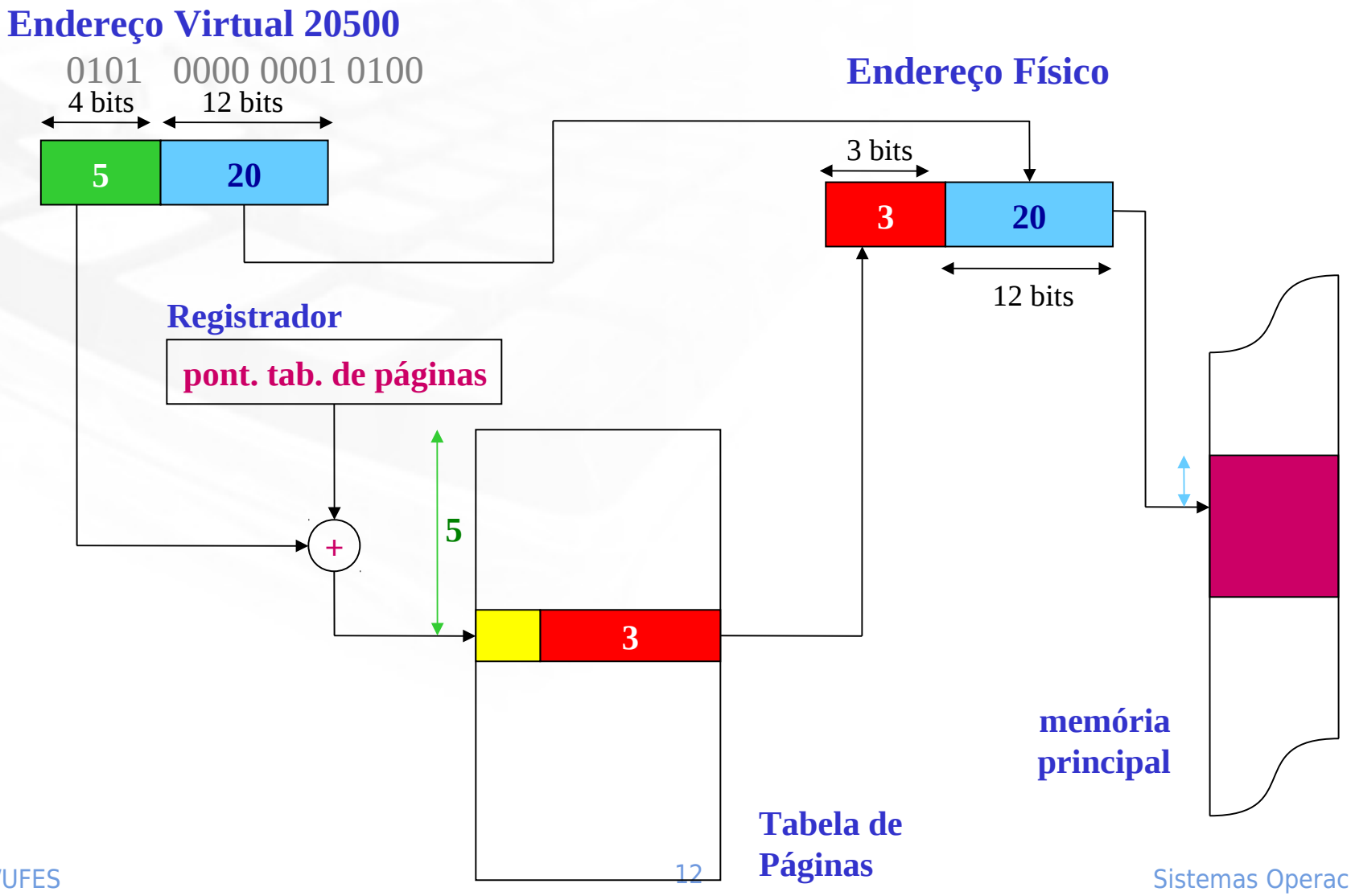
Paginação: Como funciona?

- Cada processo tem sua **Tabela de Páginas**
- Tabela de Páginas faz o mapeamento página x moldura
- O que acontece se o programa faz um acesso a uma página que não está mapeada na memória?
- Tabela de páginas pode estar só parcialmente na MP
- Dois acessos à MP

Paginação: Endereçamento

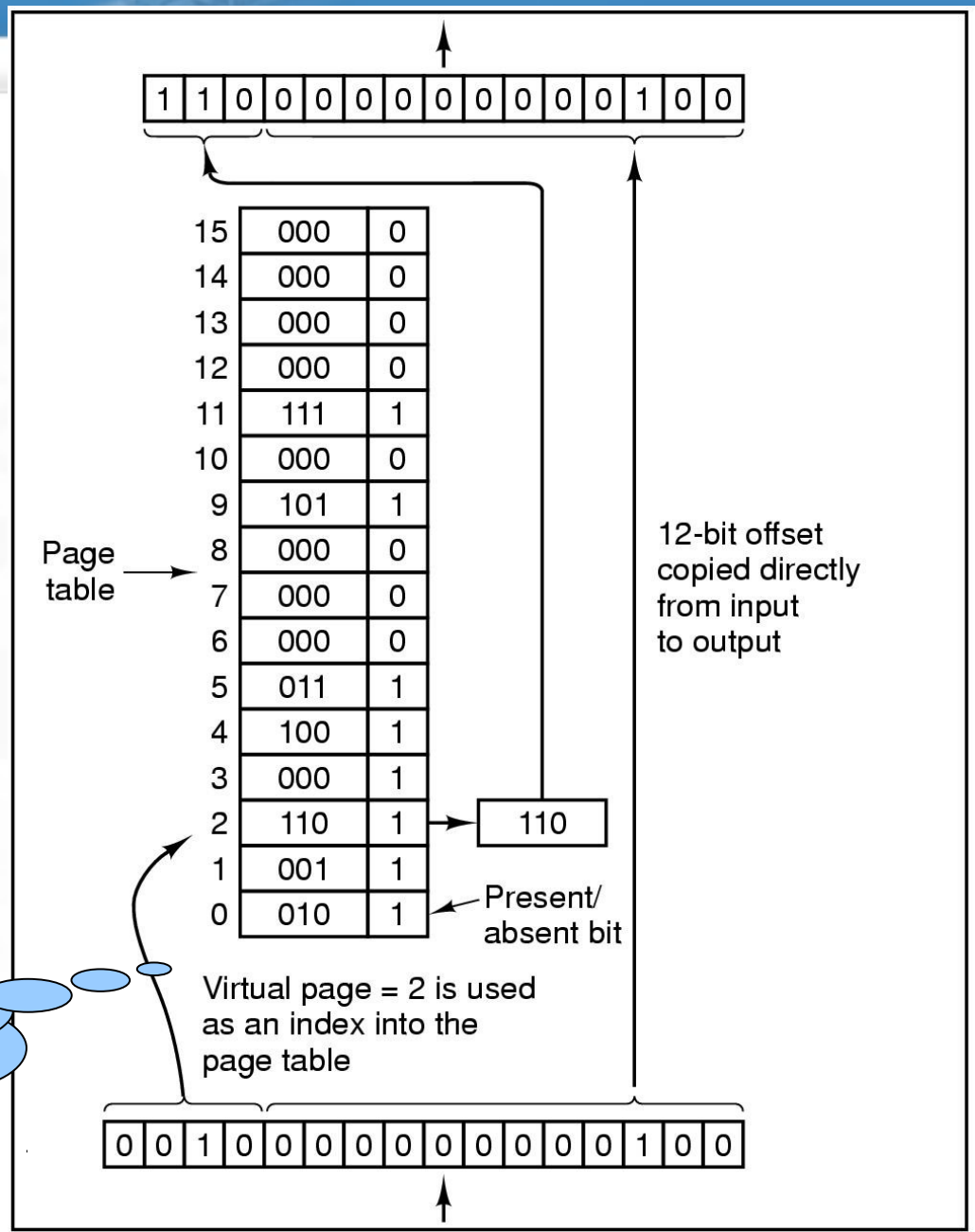


Paginação: Endereçamento – Exemplo (1)



Paginação: Endereçamento – Exemplo (2)

- Operação interna de uma MMU com 16 páginas de 4 kB

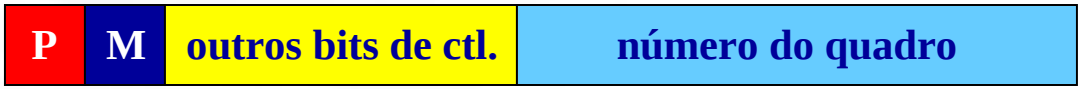


O nº da pag. é usado como índice

Paginação: Endereçamento

Endereço Virtual

Linha da Tabela de Páginas



e.g., referenciada, proteção, compartilhamento, desabilita colocação na *cache*, etc.

Paginação: Como funciona?

- O que acontece se o programa faz um acesso a uma página que não está mapeada na memória?
  - Ocorre uma *Page Fault* => a MMU força uma interrupção
- Ação do S.O.
  - Escolher uma página pouco usada, que encontra-se em alguma moldura da memória principal
    - Salvar esta página no disco (caso ela tenha sido modificada)
  - Carregar a página virtual referenciada pela instrução na moldura recém liberada
  - Atualizar o mapeamento da tabela de páginas e reiniciar a instrução causadora da interrupção

Tabela de Páginas (1)

- Problemas
  - Ela pode ser muito grande
    - Suponha uma máquina de 32 bits, 4k por página

      $\frac{2^{32} \text{ endereços virtuais}}{4k} = \frac{2^{20} \text{ entradas na tabela de páginas}}{2 \times 2^{10}}$

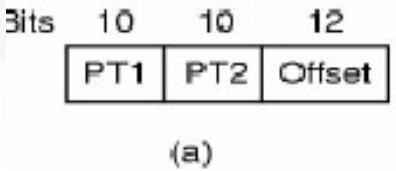
    - E uma máquina de 64bits !?!
    - Deve-se utilizar mecanismos para diminuir o tamanho da tabela
  - O mapeamento deve ser rápido
    - Mapeamento para buscar a instrução na memória
    - Instruções podem conter operandos que também encontram-se na memória

Tabela de Página Multinível (1)

- O objetivo é evitar manter toda a tabela de páginas na memória durante todo o tempo
- Apresenta-se como uma solução para o dimensionamento da tabela de páginas
- Uso de dois apontadores e um deslocamento
- Exemplo: Tabela de dois níveis
  - O endereço de 32 bits de endereço divido em 3 campos
    - PT1 [10 bits] : indexa o primeiro nível da tabela
    - PT2 [10 bits] : indexa o segundo nível da tabela
    - Deslocamento [12 bits]: => paginas de 4 KB

Tabela de Página Multinível (2)

- 1º nível com 1024 entradas
- Cada uma dessas entradas representa 4 MB
  - 4 GB / 1024



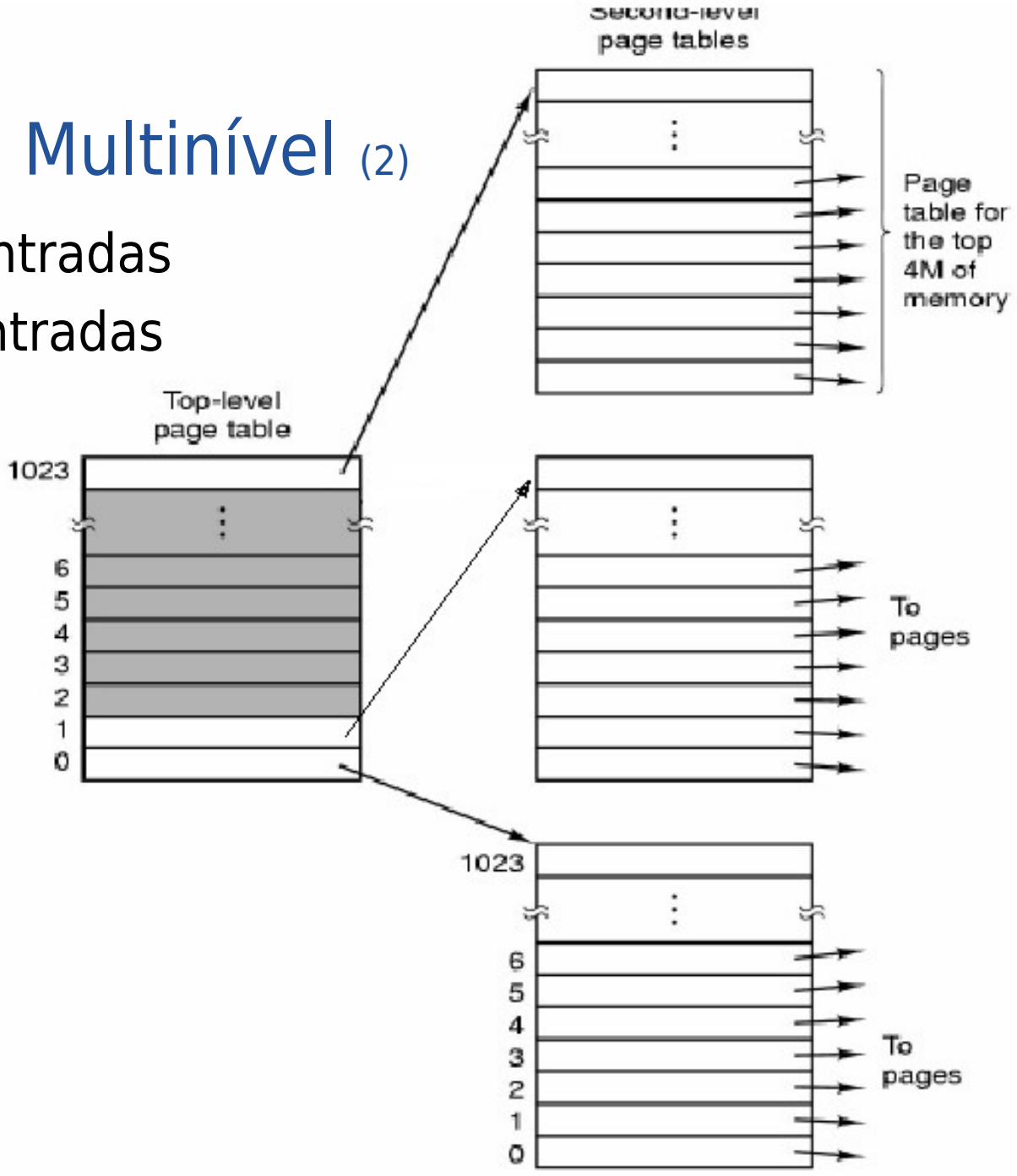
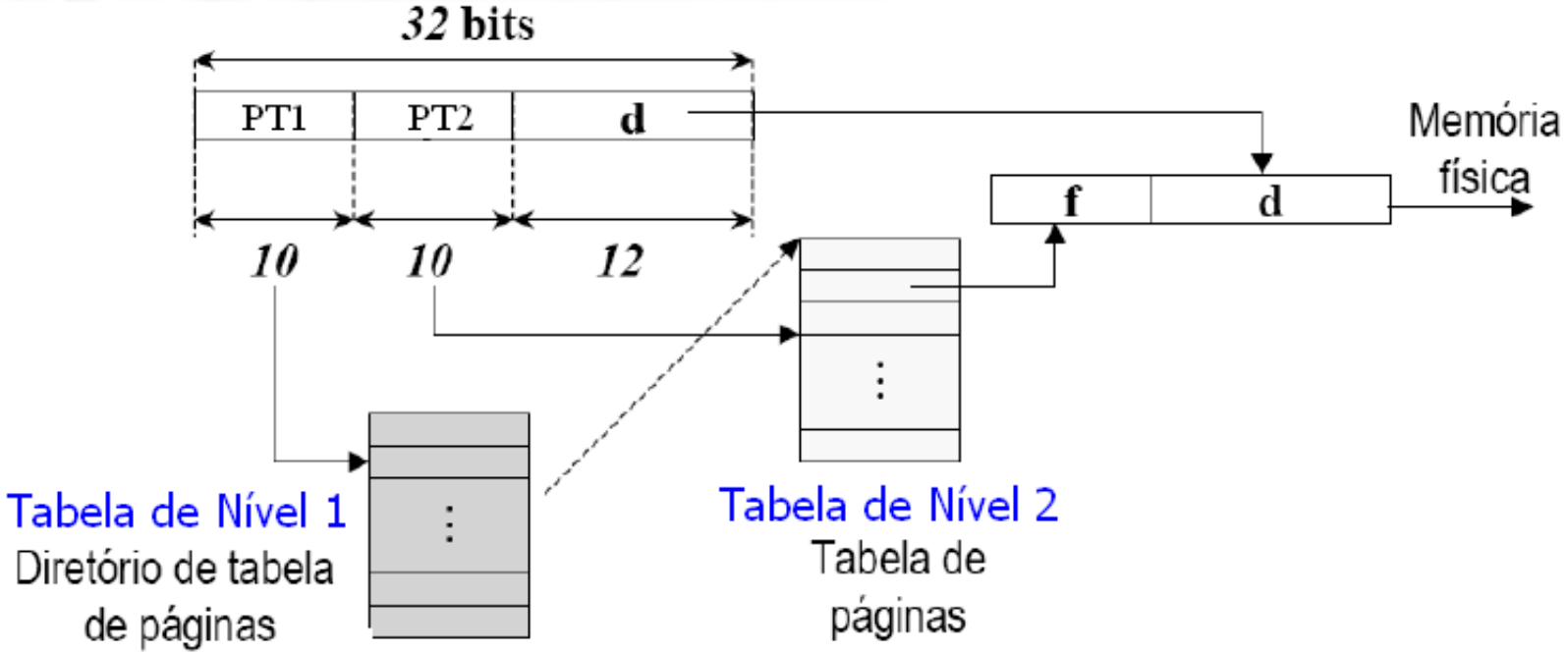


Tabela de Página Multinível (5)



- Considere o end. virtual 0x00403004 (4206596_d)
  - Qual será o endereço físico correspondente?

Tabela de Página Multinível (5)

PT1	PT2	Deslocamento
0000000001	0000000011	0000 0000 0100

- PT1: Entrada 1 da tabela do 1º nível
  - 2º bloco de 4M (4M a 8M de memória virtual)
- PT2: Entrada 3 da tabela do 2º nível
  - Esta entrada indica em qual moldura encontra-se esta página
  - O endereço físico do primeiro byte dessa moldura é somado ao deslocamento
    - Supondo a página encontre-se na moldura 1 (4k a 8k-1), o endereço físico correspondente será 4096 + 4 = 4100
  - OU:

Nº da moldura	Deslocamento	
0... 00001	0000 0000 0100	= 4100_d

Referências

- A. S. Tanenbaum, ''Sistemas Operacionais Modernos'', 3a. Edição, Editora Prentice-Hall, 2010.
  - Seção 3.3
- Silberschatz A. G.; Galvin P. B.; Gagne G.; ''Fundamentos de Sistemas Operacionais'', 8a. Edição, Editora LTC, 2010.
  - Seção 9.4