Prof. Ricardo Inácio Álvares e Silva

Gerenciamento de Memória

Sistemas Operacionais

Rápida Revisão

- Sistemas operacionais **multiprogramados** permitem múltiplos programas estarem em execução ao mesmo tempo
- Cada instância de um programa em execução é um **processo**
- Programas em execução são aqueles que tem **atributos** de execução salvos no **PCB**, para que possam sair e entrar em contexto sem alterar os resultados da execução
- Tipicamente os processos ficam alocados na **memória principal** do computador

Escopo da Aula

- Overlays
  - Viabilidade

- Memória Virtual
  - Conceito gerencial
  - Paginação e quadros
  - Mecanismo de mapeamento virtual -> real

Problemas com *Swapping*

- Apesar de simples e viável, *swapping* possui limitações indesejadas:
  - Limitação de espaço de memória para cada processo
    - Ao entrar no sistema, é dada uma certa quantidade fixada de memória ao processo
    - Se ele crescer dinamicamente, pode ser necessário reposicionamento, operação sempre indesejada
- Impossibilidade de executar programas maiores que a memória física disponível

  Exemplo: *um computador com endereçamento de 32 bits, capaz de enxergar 4 bilhões de palavras, porém com apenas 1 bilhão de palavras em memória física. Com swapping, o limite de um programa nesse computador é 1 bilhão*

Programas modulares

Overlays

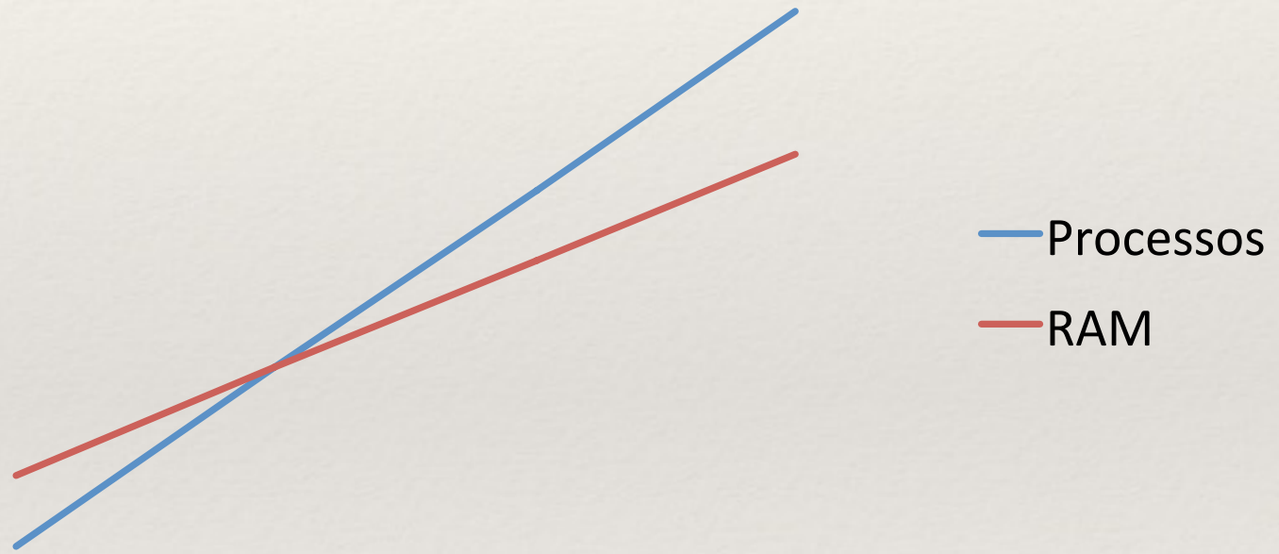
- Antes da invenção da memória virtual, um programa não poderia ser maior que a própria memória física do sistema
- Solução: organizar tais programas em módulos
  - enquanto um módulo estiver em uso, ocupa a memória principal
  - outros módulos do mesmo programa ficam no disco
  - quando um módulo finaliza, um gerenciador de módulos o coloca de volta no disco e busca o próximo módulo
- Essa técnica de programação é conhecida por ***overlays***
  - palavra que significa “transparências”, como as de retroprojetores

Problemas em *Overlays*

- Introduz complexidade na programação
  - um módulo não pode fazer referência direta a instruções, funções e dados de outros, pois estão no disco
    - se houver tal referência, irá resultar em funcionamento errático, pois a referência nunca é resolvida automaticamente, nem o sistema acusa erro
- Além disso, a separação em do programa em módulos deve ser feita manualmente, para cada programa, analisando caso a caso
- Impossibilidade de generalização, complexidade de programação:
  - Custo Alto

Prospecção de Inviabilidade

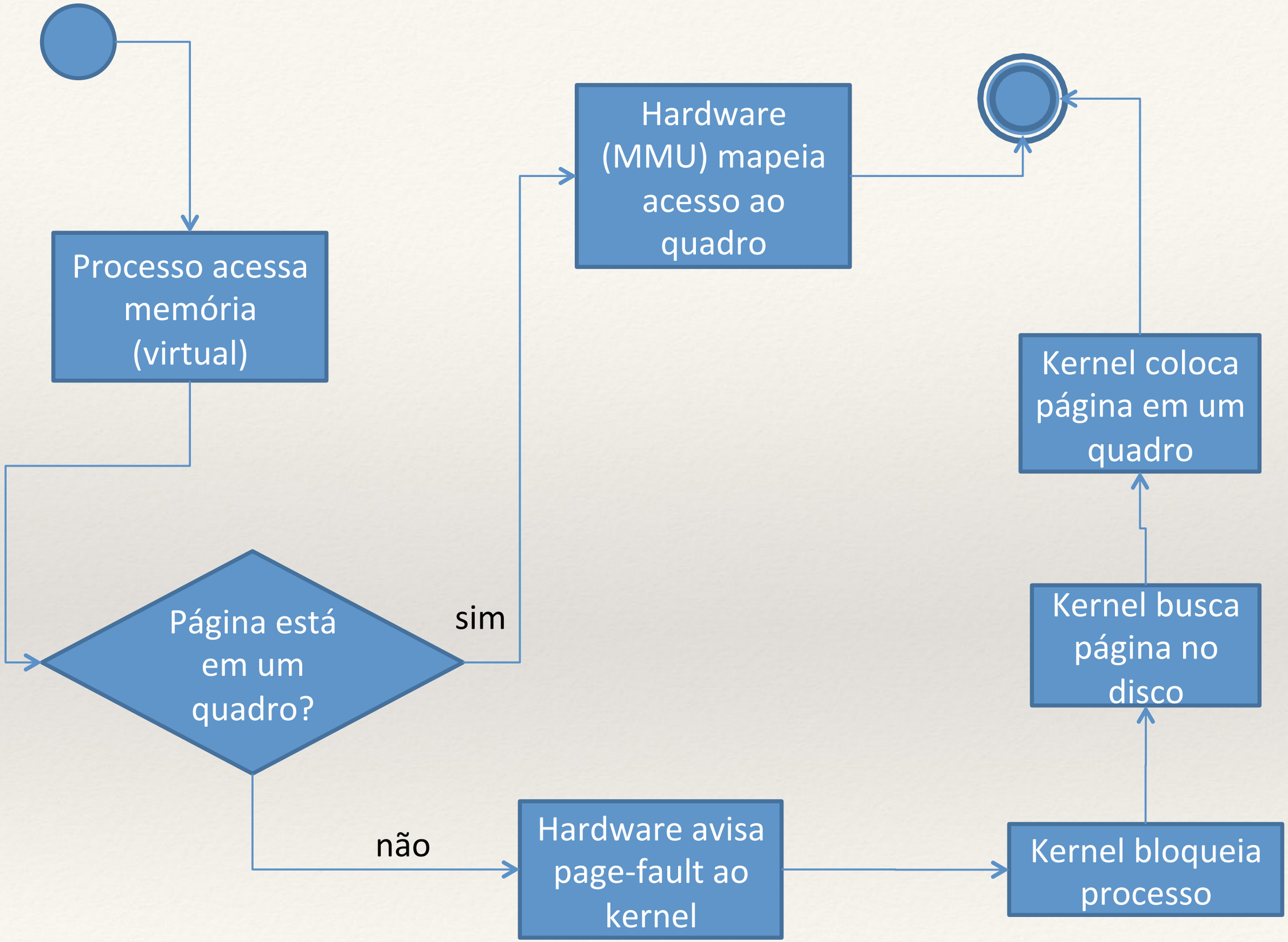
- Gráfico: quantidade de processos *e* disponibilidade de memória principal *por* tempo
- Quantidade de processos cresce mais que memória
- Necessidade de guardar programas em execução no disco



Folheando Livros

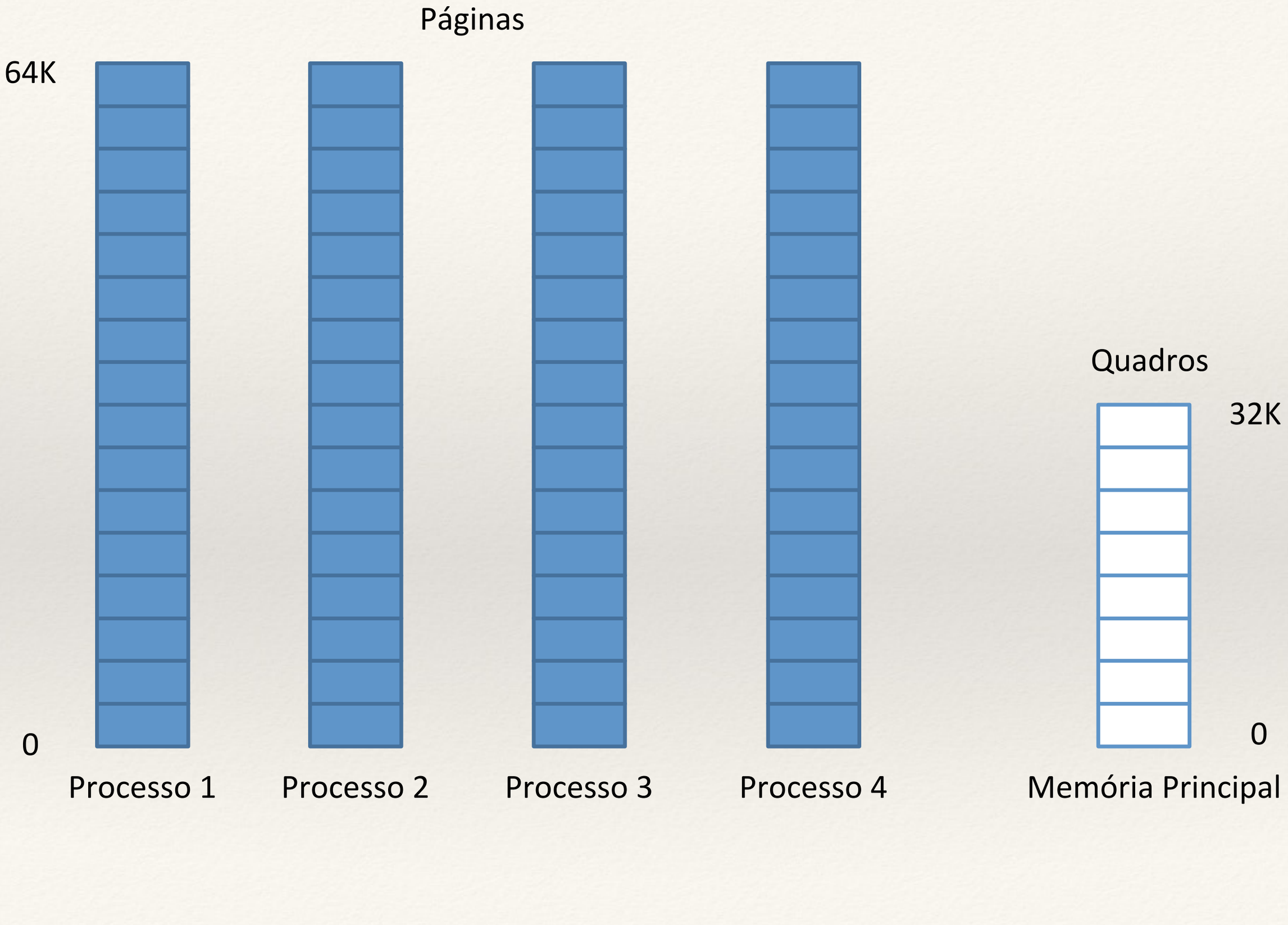
Memória Virtual

- Separa a memória real em **quadros**
  - pedaços de memória com tamanhos uniformes
- Cada processo possui seu próprio **espaço de memória virtual**, com tamanho igual ao do domínio de memória do processador
- Os processos são divididos em **páginas**, com tamanhos iguais aos dos quadros, inicialmente existentes apenas na memória persistente do computador
- Quando processo referencia um endereço de memória
  - o hardware verifica a qual página ele pertence
  - verifica se a página está disponível em algum quadro
  - se estiver, faz a tradução do endereço virtual para o real e o acessa
  - se não estiver, acontece a **falha de página** (*page-fault*), em que o hardware requisita ao *kernel* que disponibilize a página necessária em algum quadro


Processo acessa memória (virtual)
Página está em um quadro?
sim
não
Hardware (MMU) mapeia acesso ao quadro
Hardware avisa page-fault ao kernel
Kernel bloqueia processo
Kernel busca página no disco
Kernel coloca página em um quadro

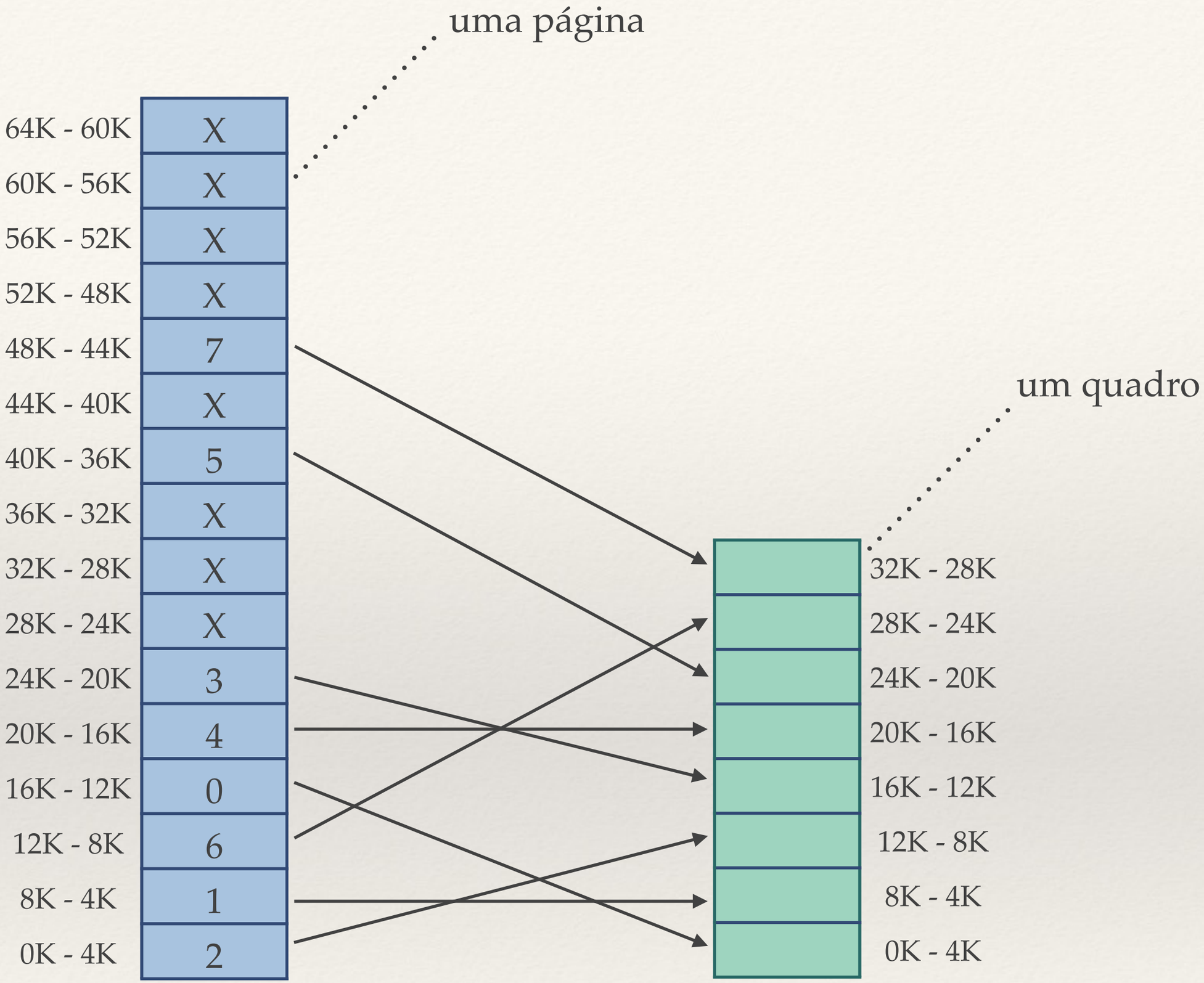
Análise de um caso concreto

- Suponha um computador com as seguintes características:
  - Endereçamento da CPU: 16 bits (máximo de 64K palavras)
  - Memória física: 32K palavras
    - Tamanho de páginas e quadros: 4K palavras
- Memória virtual desse sistema:
  - Memória de cada processo: 64K
  - Quantidade de páginas de cada processo: 16
  - Quantidade de quadros no sistema: 8


Páginas
64K
0
Processo 1
Processo 2
Processo 3
Processo 4
Quadros
32K
0
Memória Principal

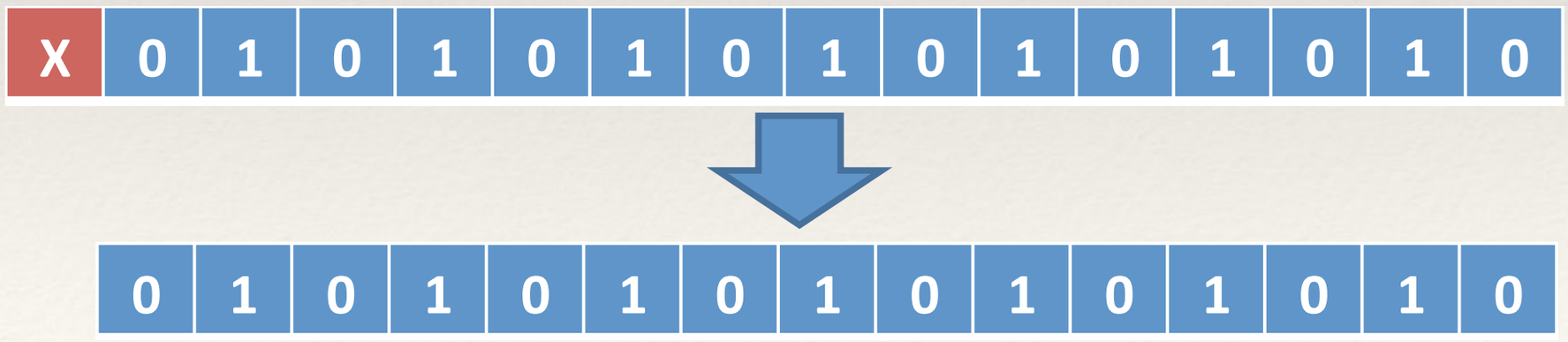
Palavras, Páginas e Quadros

- Os processos acreditam que
  - o sistema tem o máximo de memória possível
  - a memória é inteira deles
- Diferentemente do *swapping*, eles de fato podem acessar a memória inteira, não há limite superior
- Cada posição da memória pertence a uma página, assim como letras em um livro
- Quando um endereço em uma página é acessada, esta é colocada em algum quadro disponível da memória física
- Qualquer página de qualquer processo pode ser aquinhoada em qualquer quadro


uma página
64K - 60K
60K - 56K
56K - 52K
52K - 48K
48K - 44K
44K - 40K
40K - 36K
36K - 32K
32K - 28K
28K - 24K
24K - 20K
20K - 16K
16K - 12K
12K - 8K
8K - 4K
0K - 4K
X
X
X
X
7
X
5
X
X
X
3
4
0
6
1
2
um quadro
32K - 28K
28K - 24K
24K - 20K
20K - 16K
16K - 12K
12K - 8K
8K - 4K
0K - 4K
Memória virtual
de um processo
Memória física
do processador

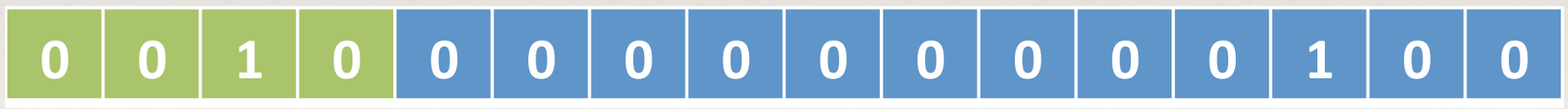
- Como uma página de um processo pode estar em qualquer quadro, sem ordem, e ainda assim o acesso ser correto?
  - *vamos entender o endereçamento à memória no nível binário*
- Suponha um processador com 16 bits de endereçamento
  - utiliza registrador de 16 bits para referenciar posições da memória:

- Pode acessar até 2^16 = 64K posições
  - porém, se ele tiver apenas 32K de memória física, acessa 2^15:



- Mesmo havendo apenas 32K de memória, cada processo utiliza todos os 16 bits para o acesso
- Suponha que um processo acesse a posição 8196:
  - 8196d = 0010000000000100b

- Como cada página possui 4K, e 4K = 2^12, os 12 primeiros bits, indicam qual a posição da memória nesta página
- Cada processo possui 16 páginas, 16 = 2^4, logo os últimos 4 bits indicam a página daquele acesso:



Página
0010b = 2d

Posição nesta página
000000000100b = 4d

- Este endereço, 8196, acessado por um processo, é denominado **endereço virtual**
- O endereço virtual precisa ser traduzido em uma posição de **endereço real**

- A região em azul, que representa a posição da memória na página, já está correta, não precisa modificar
- A parte verde, representa qual página foi acessada
  - este valor precisa ser traduzido para o quadro em que a página se encontra

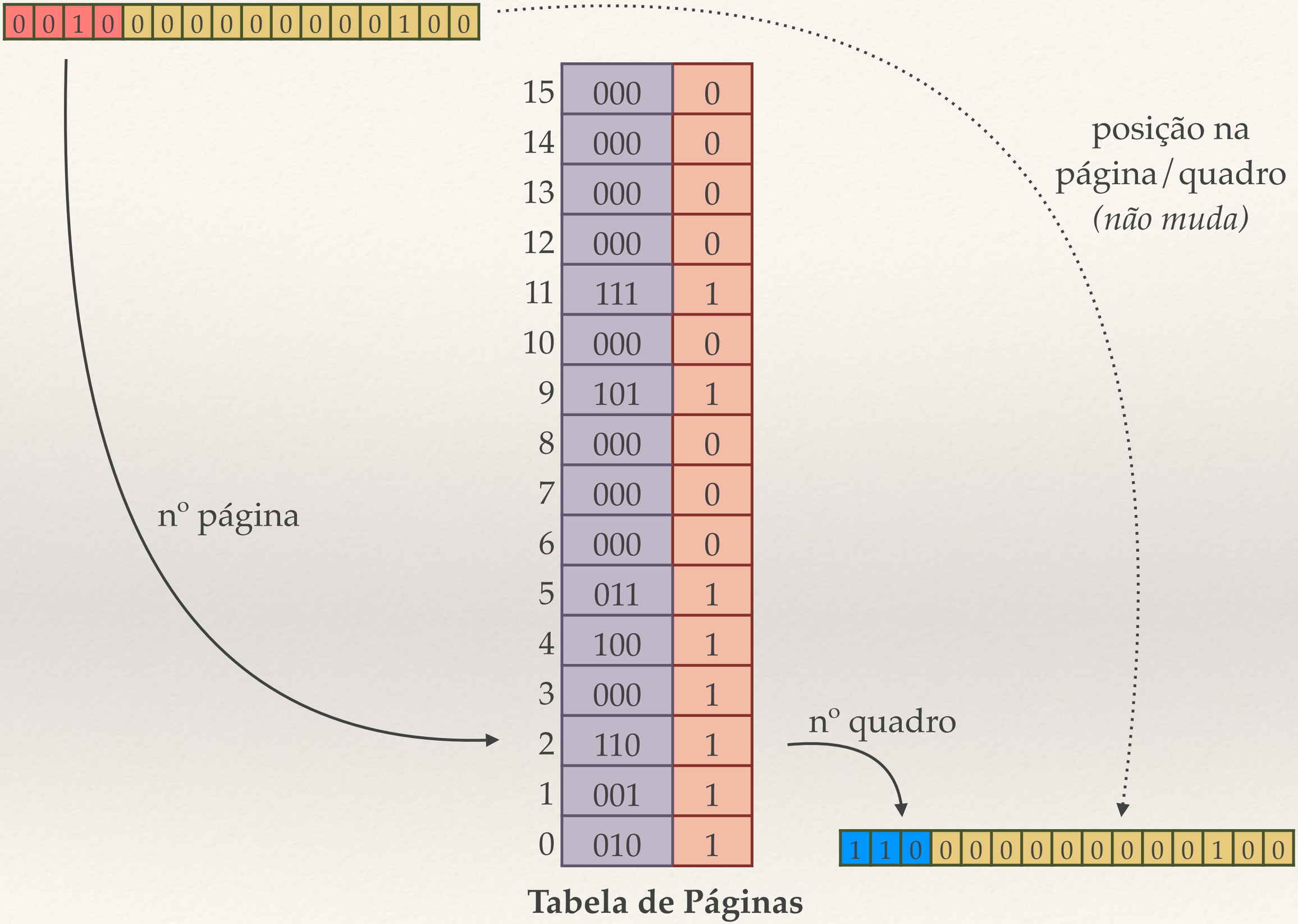


Tabela de Páginas

- Essa tabela de páginas fica em uma memória interna da MMU, registradores de tradução
- Cada processo possui sua própria tabela de páginas
- Quando um processo sai de contexto, a sua tabela de páginas é guardada na memória
- Quando um processo entra em execução, sua tabela de páginas é carregada da memória para a MMU