

Aula 13

Memória Virtual

Prof. Omar Paranaiba Vilela Neto



Memória Virtual

:: Introdução

- ❖ Memória Virtual é a técnica que dá ao programador a ilusão de poder acessar rapidamente um grande espaço de endereçamento
- ❖ Objetivos da técnica:
 - ▶ Permitir que haja um meio seguro e eficiente de se compartilhar informações, armazenadas na memória, entre vários programas
 - ▶ Minimizar os problemas causados aos programas pela existência de uma pequena quantidade de memória principal

Memória Virtual

:: Introdução

- ❖ Os programas que compartilham a memória de determinada máquina mudam dinamicamente durante o processo de execução
- ❖ Cada programa deve ser compilado usando seu próprio espaço de endereçamento (ou seja, em uma região da memória acessível somente a esse programa)
- ❖ A técnica de memória virtual realiza a tradução do espaço de endereçamento de um programa para seus endereços reais

Memória Virtual

:: Introdução

- ❖ A técnica de memória virtual permite que o tamanho de um único programa **exceda a quantidade total de memória real disponível para sua execução**
- ❖ A técnica de memória virtual **gerencia automaticamente os seguintes** dois **níveis** de hierarquia:
 - ▶ memória principal (física)
 - ▶ memória secundária
- ❖ **Antigamente**, os programas eram divididos em **pedaços**
 - ▶ Os pedaços **mutuamente exclusivos** (overlays) eram **identificados**
 - ▶ Carga/exclusão de overlays da memória era realizada **sob controle do próprio programa**

Memória Virtual

:: Introdução

- ❖ A memória virtual faz com que a **memória principal funcione como uma cache** para memória secundária (discos magnéticos)
- ❖ Vantagens:
 - ▶ Ilusão de ter **mais memória física**
 - ▶ Realocação de programa
 - ▶ Proteção entre processos
 - ▶ Separação entre memória lógica e memória física:
 - Memória **Lógica**: visão de um **processo**
 - Memória **Física**: visão do **processador**

Memória Virtual × Memória Cache

Memória
Virtual

Página ou Segmento

Falta de página

Substituição de erros
controlado pelo Sistema
Operacional

Tamanho determinado pelo
tamanho do endereço do
processador

Divide espaço do
armazenamento secundário
com sistema de arquivos

Memória
cache

Bloco

Miss (falha)

Substituição de erro
controlada pelo
hardware

Independente do tamanho
do endereço do
processador

Espaço totalmente
utilizado como
memória

Memória Virtual × Memória Cache

Parâmetro	Cache L1	Memória virtual
Tamanho do bloco (página)	16-128 bytes	4.096-65.536 bytes
Tempo de acerto (hit)	1-3 ciclos	100-200 ciclos
Penalidade de falha (Acesso)	8-200 ciclos (6-160 ciclos)	1 – 10 M ciclos 0,8 – 8 M ciclos
Penalidade de falha (Transferência)	(2-40 ciclos)	0,2 – 2 M ciclos
Taxa de falha	0,1-10%	0,00001-0,001%
Mapeamento de endereços	25-45 bits de endereço físico para 14-20 bits de endereço de cache	32-64 bits de endereços virtuais para 25-45 bits de endereços físicos

Memória Virtual

:: Conceitos

Página

- bloco de bytes de **tamanho fixo**

Segmento

- bloco de bytes de **tamanho variável**

Falta de página

- ocorre quando uma **página** acessada **não** está **presente** na **memória principal**

Endereço físico (ou real)

- um **endereço da memória principal**

Endereço virtual

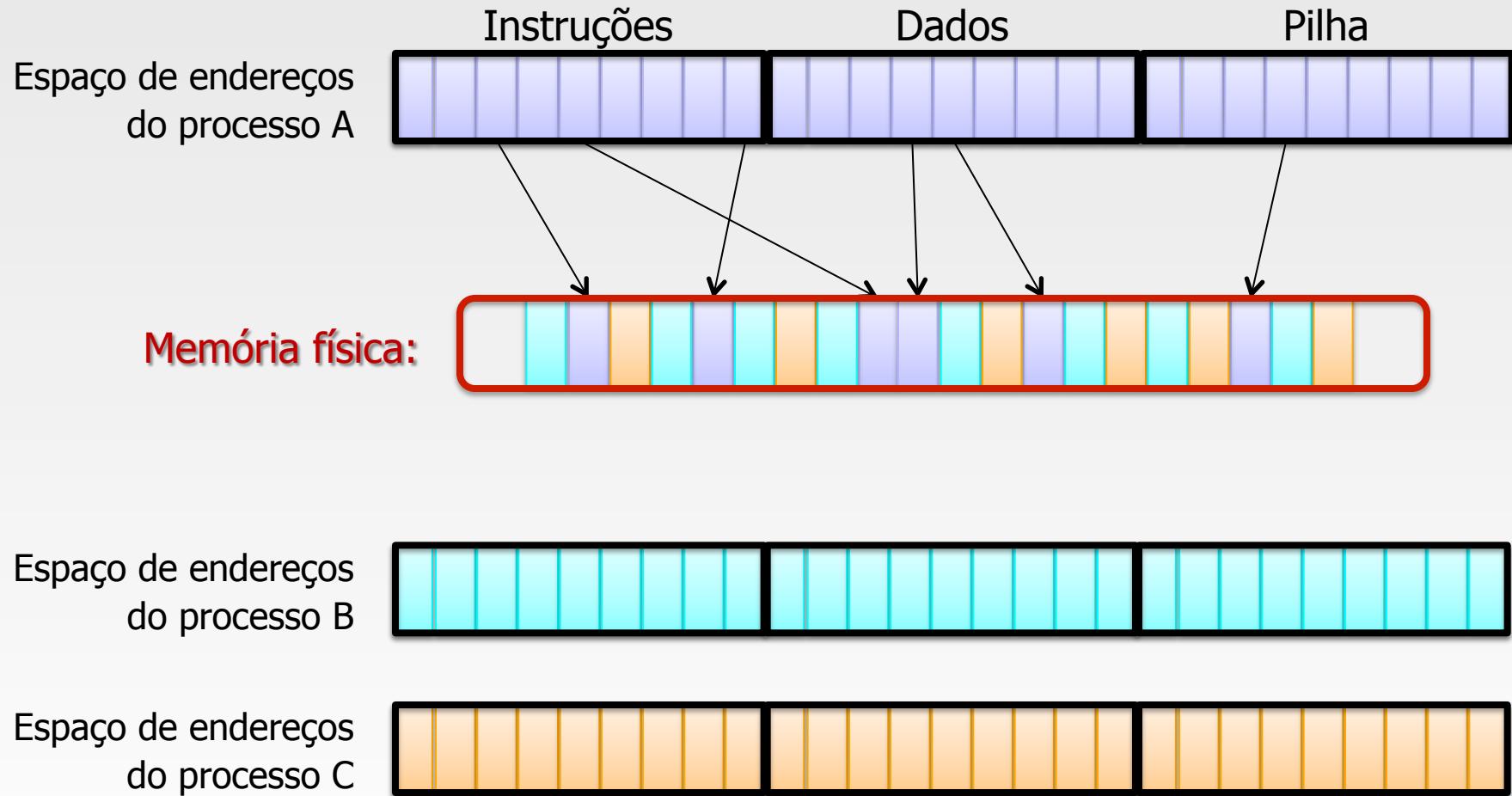
- **endereço que corresponde** a um local no **espaço virtual**

Tradução de endereço (ou mapeamento)

- **processo de tradução** de um endereço **virtual** para um endereço **físico** quando a memória principal é acessada

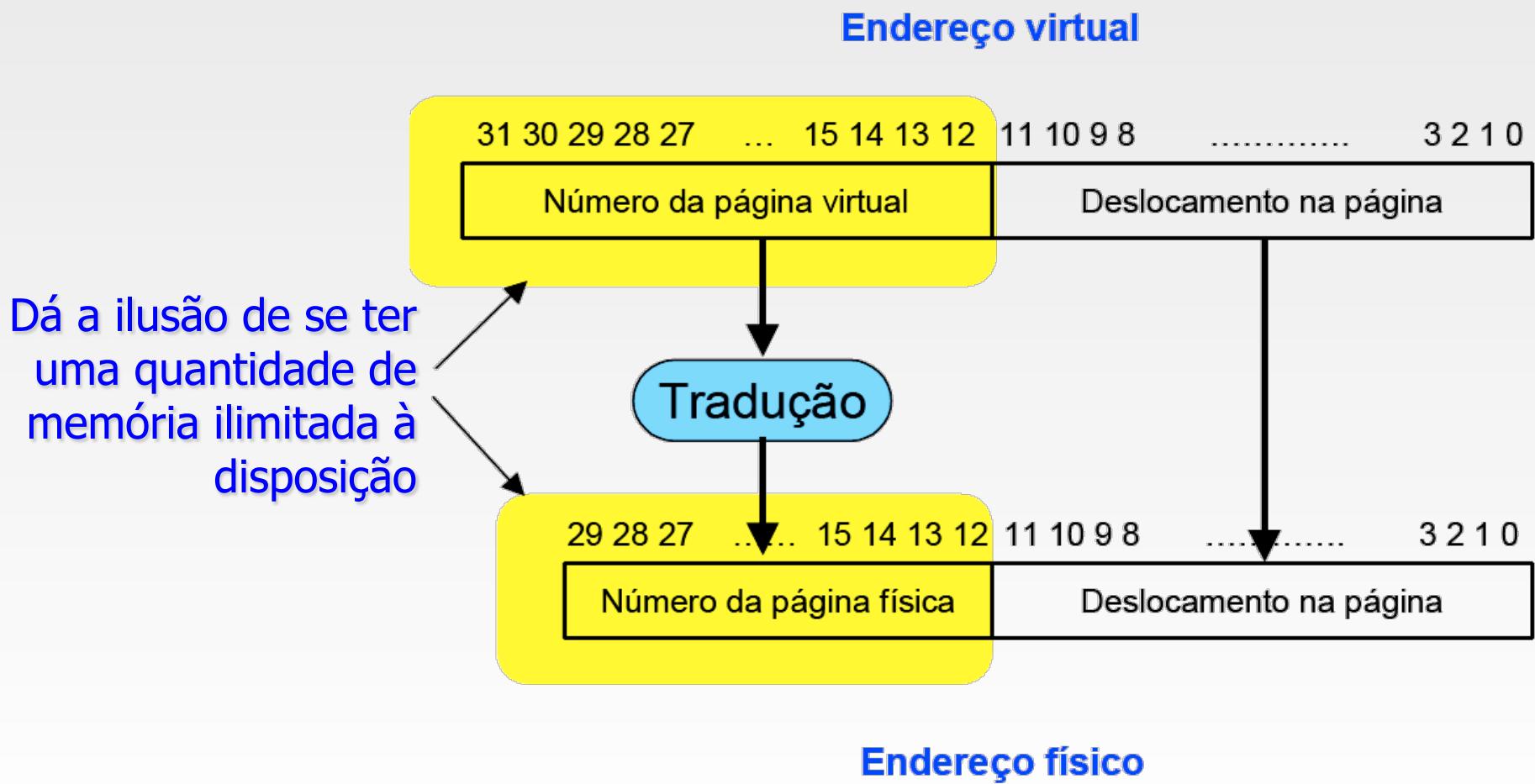
Memória Virtual

:: Tradução de endereços



Memória Virtual

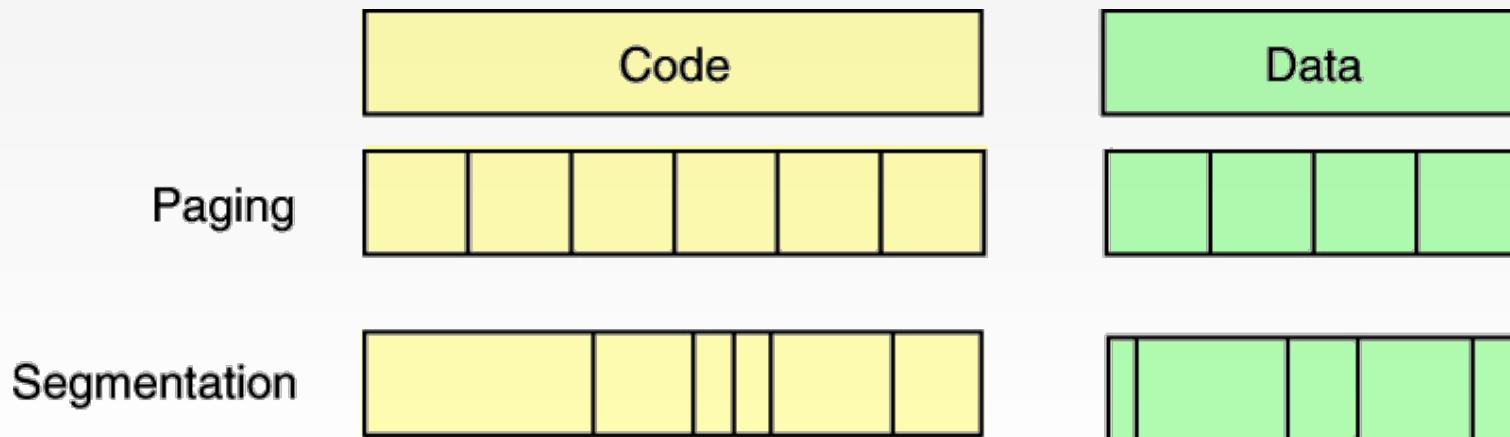
:: Tradução de endereços



Memória Virtual

:: Página × Segmento

- ❖ A paginação utiliza blocos de tamanho fixo
- ❖ A segmentação utiliza blocos de tamanho variável
- ❖ Um segmento consiste de duas partes:
 - ▶ Número de segmento
 - ▶ Offset de segmento



Memória Virtual

:: Página × Segmento

❖ Por que utilizar segmentação?

- ▶ Suporte a **métodos de proteção** mais avançados
- ▶ **Compartilhar** um espaço de endereçamento

❖ Desvantagem:

- ▶ **Divide o espaço de endereço** em partes logicamente separadas, que precisam ser manipuladas como um endereço de duas partes

Memória Virtual

:: Página × Segmento

- ❖ A decisão de usar memória virtual **paginada** ou memória virtual **segmentada** afeta o desempenho da CPU

	Página	Segmento
Palavras por endereço	Uma	Duas (segmento e offset)
Visível ao programador?	Invisível ao programador de aplicação	Pode ser visível ao programador de aplicação
Substituição de blocos	Trivial (todos os blocos são do mesmo tamanho)	Difícil (deve encontrar porções contínuas e sem uso da memória principal)
Eficiência de uso de memória	Fragmentação interna (porção não usada da página)	Fragmentação externa (porções não usadas na memória principal)
Eficiência de tráfego de disco	Sim (ajuste do tamanho de página para balancear tempo de acesso e de transferência)	Nem sempre (pequenos segmentos podem transferir poucos bytes apenas)

Quatro Perguntas Básicas sobre a Hierarquia de Memória

1. **Posicionamento da página:** Onde a página deve ser colocada na memória principal?
2. **Identificação da página:** Como a página é encontrada na memória principal?
3. **Substituição de página:** Quais páginas serão trocadas em uma falta?
4. **Estratégia de gravação:** O que acontece em uma escrita de página?

P1. Posicionamento da página

- ❖ A penalidade de erro para a memória virtual é muito alta, pois envolve o acesso a um dispositivo de armazenamento magnético rotativo
- ❖ Em razão disso, para reduzir a frequência de faltas de páginas, os sistemas operacionais utilizam o esquema de posicionamento totalmente associativo

P2. Identificação da página

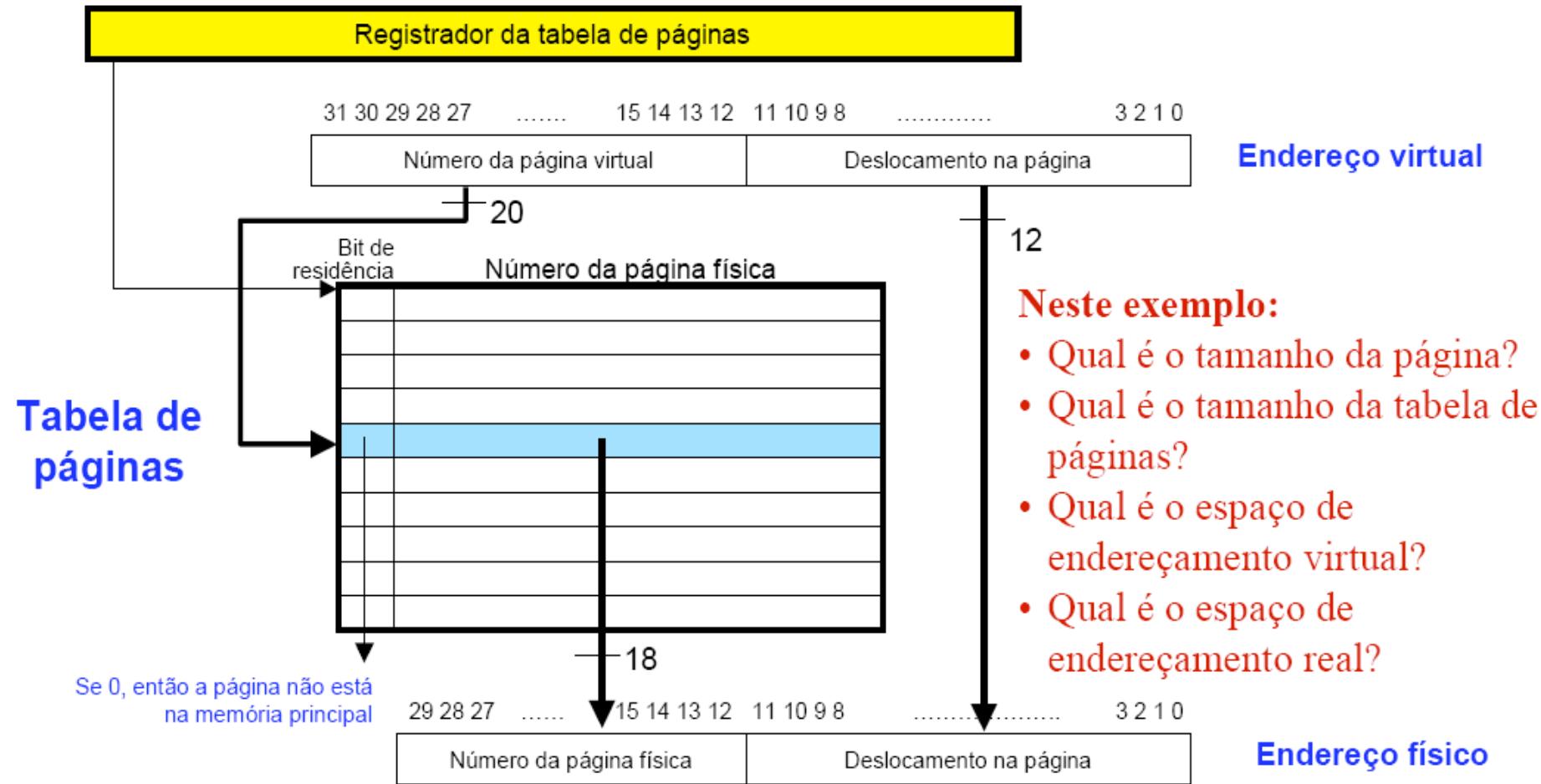
- ❖ A **desvantagem** da escolha do posicionamento **totalmente associativo** está em **localizar uma entrada**, já que ela pode estar em **qualquer lugar** da memória virtual
- ❖ Como o **espaço** ocupado pela memória virtual é **maior** que aquele ocupado pela memória cache, o **tempo de busca** aumenta substancialmente
- ❖ Para contornar essa desvantagem, utiliza-se uma **tabela de páginas**, uma estrutura que **indexa** as **traduções** de endereços **virtuais** para endereços **físicos**

P2. Identificação da página :: Tabela de Páginas

- ❖ A tabela de páginas está armazenada na memória principal
- ❖ É indexada com o número da página extraído do endereço virtual e contém o número da página física correspondente
- ❖ Como cada página virtual indexa uma entrada da tabela de páginas, nenhuma tag é necessária

P2. Identificação da página

:: Tabela de Páginas



P2. Identificação da página :: Tabela de Páginas

- ❖ Para que mais um programa utilize alternadamente o processador, é necessário salvar seu estado:
 - ▶ Registradores da CPU
 - ▶ Contador de programa (PC)
 - ▶ Tabela de páginas
- ❖ Em vez de se salvar a tabela de páginas inteira, o SO conserva apenas o registrador da tabela de páginas:
 - ▶ Componente do HW de memória que aponta a posição inicial da tabela de páginas na memória principal
- ❖ Cada programa possui sua própria tabela de páginas

P2. Identificação da página :: Tamanho da Tabela de Páginas

❖ Considere:

- ▶ Espaço de endereçamento virtual: 48 bits
- ▶ Páginas de 4KB
- ▶ 4 bytes por entrada da tabela de páginas (18 bits + controle)

❖ Número de entradas da tabela de páginas:

$$\frac{2^{48}}{2^{12}} = 2^{36}$$

❖ Tamanho da tabela de páginas:

$$4 \times 2^{36} = 2^{38} = 256 \text{ GB}$$

P2. Identificação da página :: Tamanho da Tabela de Páginas

- ❖ Técnicas para reduzir tamanho da tabela de páginas:
 1. Manter um **registraror que limite o tamanho** da tabela de páginas para um determinado processo
 2. Usar o **número da página física** (em vez da virtual) como entrada da tabela e aplicar uma função de hashing para realizar a inversão
 3. Utilizar **múltiplos níveis** de mapeamento
 4. Permitir o **paginamento** da tabela de páginas

P3. Substituição de página

:: Faltas de Páginas

- ❖ Bit de **validade** (residência) = 0 indica **falta de página**
 - ▶ Nesse caso, o **SO assume o controle**, por meio do mecanismo de exceção
- ❖ O sistema operacional precisa:
 - ▶ **Encontrar a página faltante** no nível hierárquico inferior (geralmente, no HD)
 - ▶ **Decidir em que lugar da memória principal deve ser colocada** a página requisitada
- ❖ O endereço virtual não informa em que posição do HD está a página que gerou a falta de página

P3. Substituição de página

:: Faltas de Páginas

- ❖ O SO cria **espaço em disco** para todas as páginas virtuais de um processo, quando da criação do processo
 - ▶ Tal espaço é conhecido como **área de swap**
- ❖ Nesse momento, o **SO** também **cria uma estrutura** de dados para **controlar onde** cada **página virtual** **está guardada** no disco
- ❖ Tal estrutura pode ser:
 - ▶ Parte da **tabela de páginas**, ou
 - ▶ **Estrutura auxiliar**, indexada da mesma forma que a tabela de páginas

P3. Substituição de página

- ❖ Em uma memória totalmente associativa, todos os blocos são candidatos à substituição

Estratégias para a substituição de blocos:

Aleatória: os blocos candidatos à substituição são escolhidos ao acaso, possivelmente contando com algum auxílio de hardware

Bloco menos usado recentemente (LRU): o bloco substituído é aquele menos utilizado recentemente

P4. Estratégia de gravação

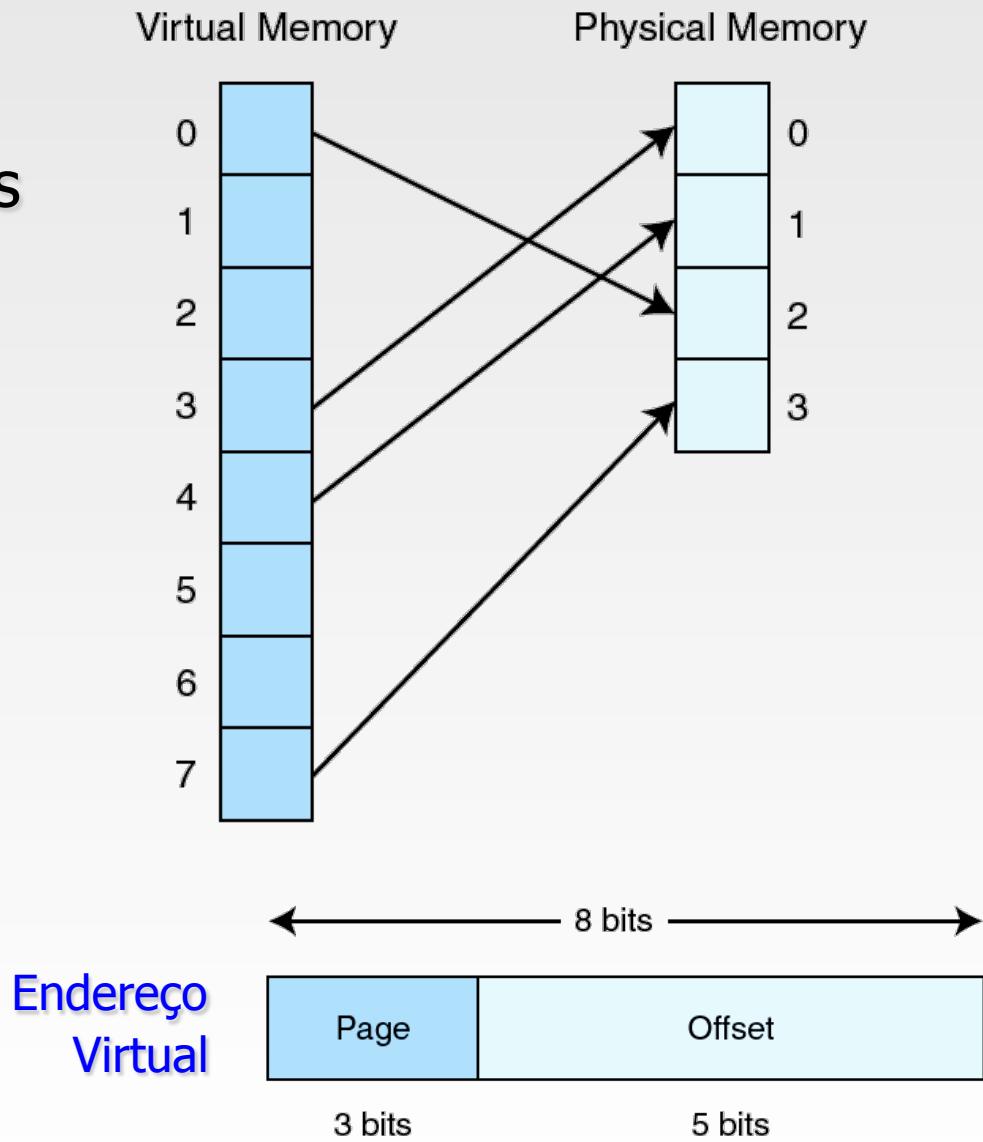
- ❖ A escrita no **disco** consome de **1 a 10 milhões de ciclos** de clock
- ❖ Esquema **write-through** não funciona para memória virtual
 - Mesmo usando um **buffer de escrita**, este deveria ser **muito maior** que aquele utilizado em memórias cache, o que encareceria o hardware
- ❖ Esquema **write-back** é usado:
 - Página é copiada para o disco no momento em que for substituída (nomenclatura: “**copy-back**”)

Memória Virtual

:: Exemplo

Considere:

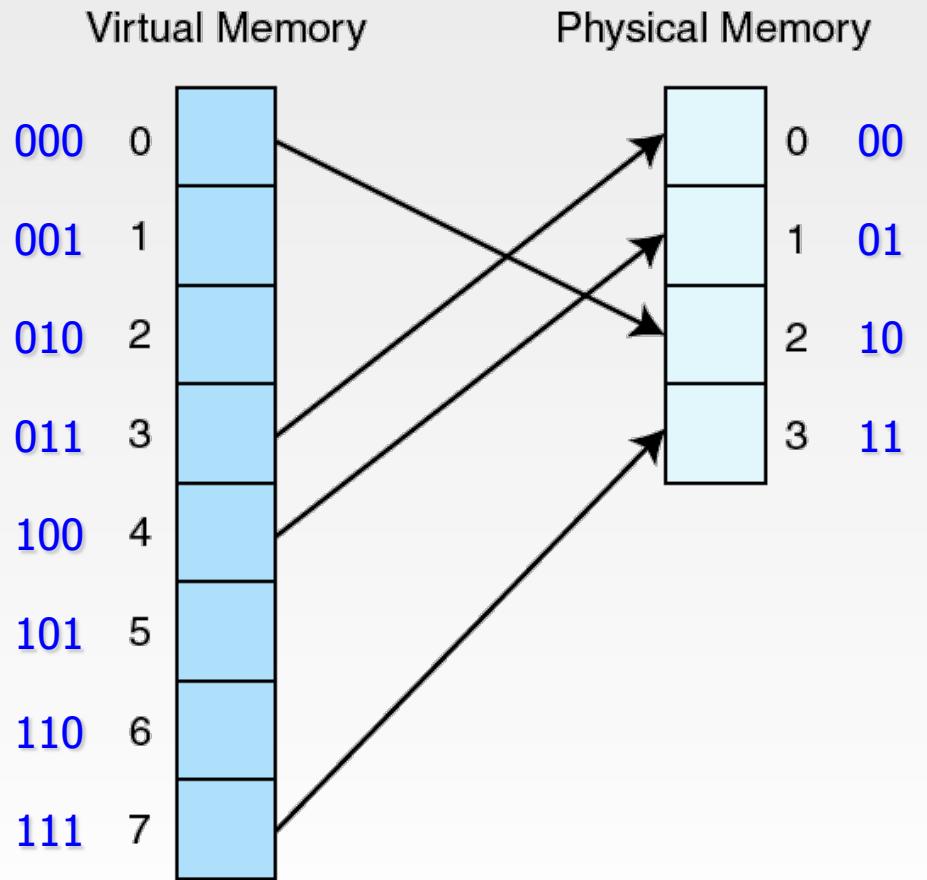
- ❖ Páginas de 32 (2^5) words
- ❖ Memória virtual de 256 (2^8) words:
 - ◆ 8 páginas de 32 words
 - ◆ Endereços de 8 bits
- ❖ Memória física de 128 (2^7) words:
 - ◆ 4 páginas de 32 words
 - ◆ Endereços de 7 bits



Memória Virtual

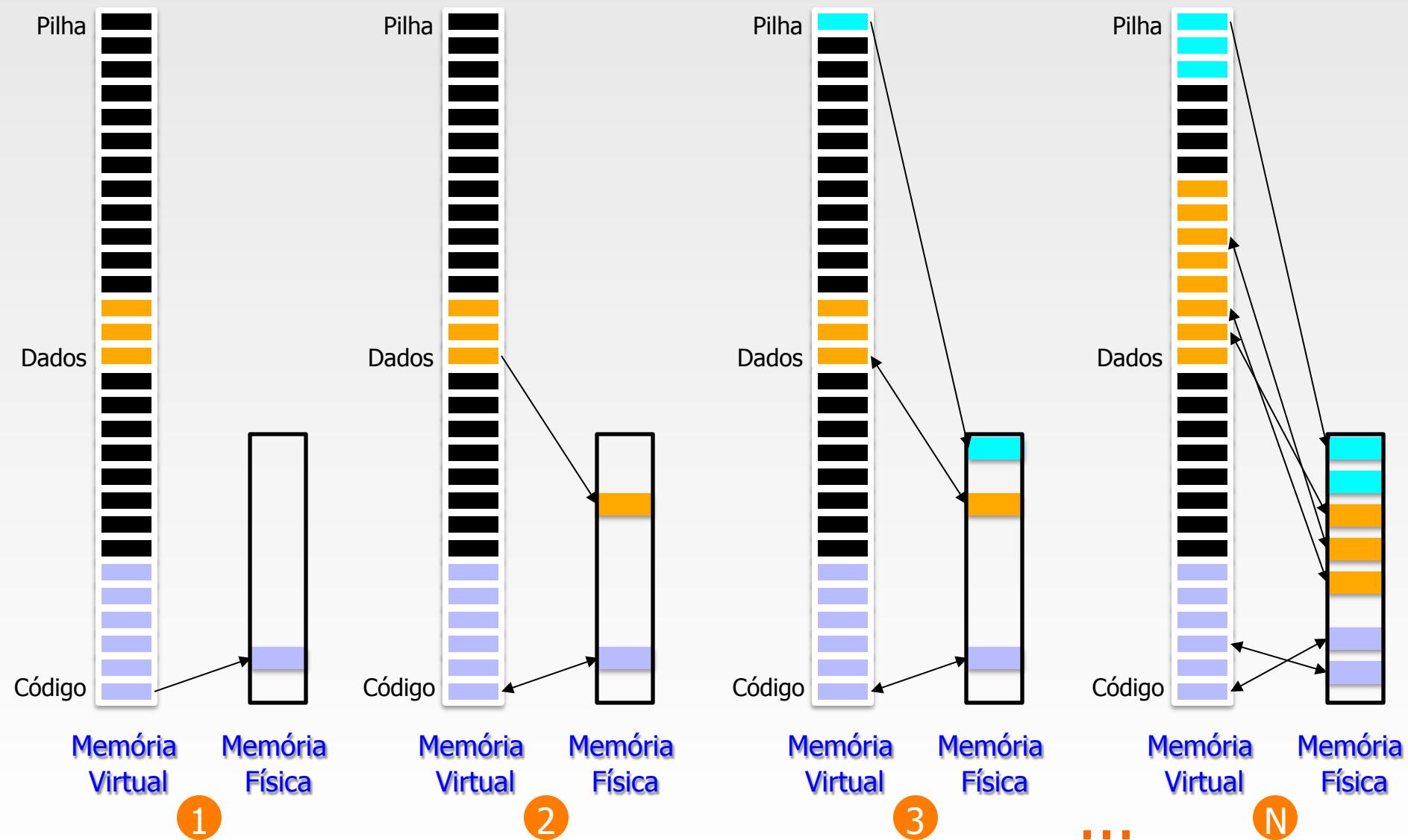
:: Exemplo

- ❖ Montando uma tabela de páginas:



Página	Quadro	Bit de validade
000	10	1
001	-	0
010	-	0
011	00	1
100	01	1
101	-	0
110	-	0
111	11	1

Paginação sob demanda

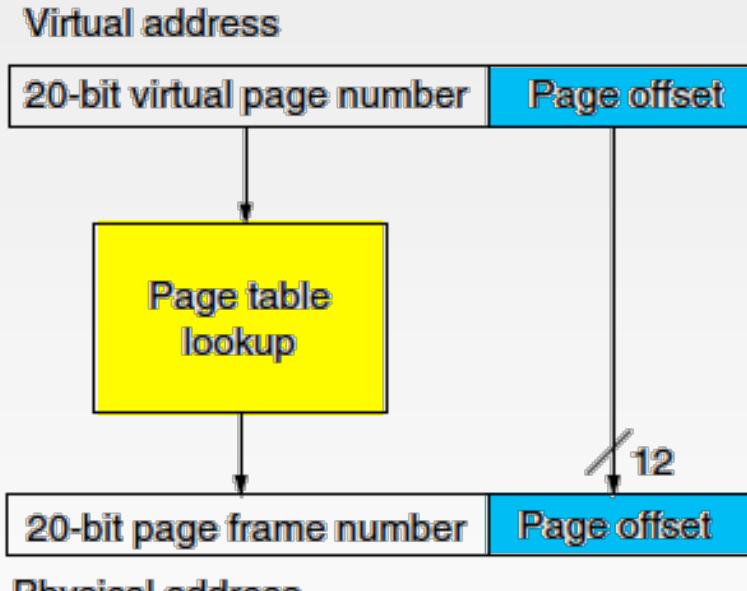


TLB – Translation Lookaside Buffer

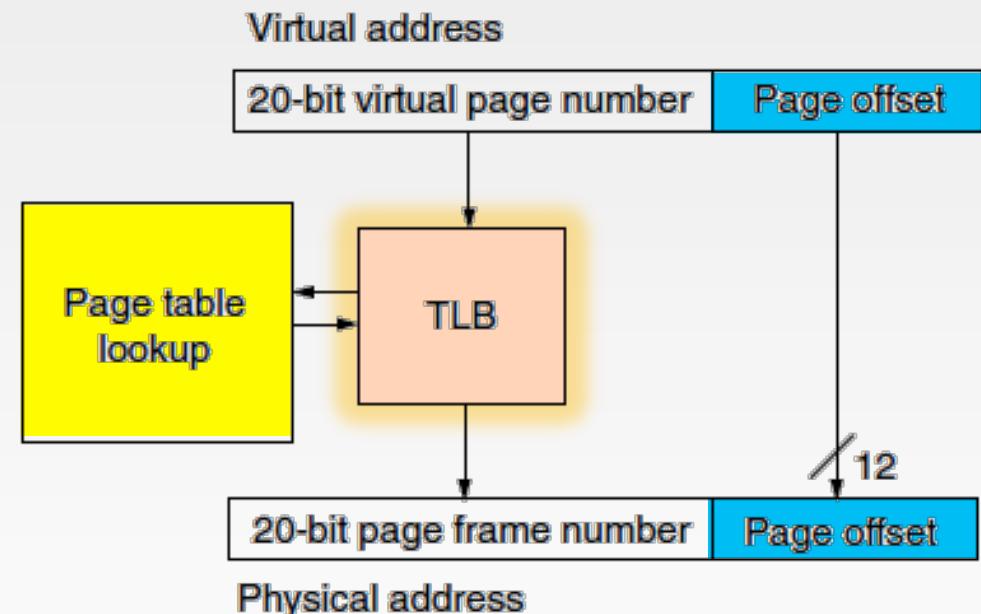
- ❖ Cada acesso à memória exige, na verdade, **dois acessos à memória principal**:
 - ▶ Um para obter o endereço físico (consulta à tabela de páginas)
 - ▶ Outro para buscar a informação
- ❖ Uma alternativa para **melhorar o desempenho** é lembrar das últimas conversões de endereços, baseando-se no **princípio da localidade**
- ❖ As **traduções mais recentes** são guardadas em uma memória cache especial: **Translation lookaside buffer (TLB)**

TLB – Translation Lookaside Buffer

- ❖ A TLB é uma memória cache, de 16 a 512 entradas, acessada pelo número de página virtual



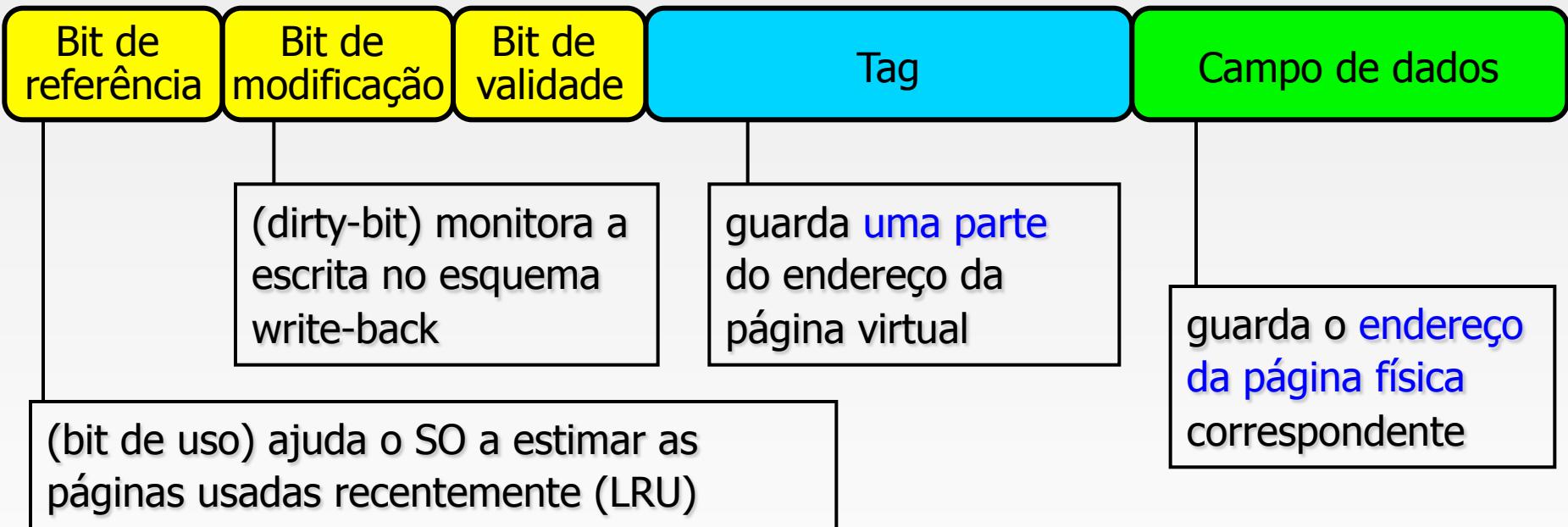
Tradução de endereços
sem TLB



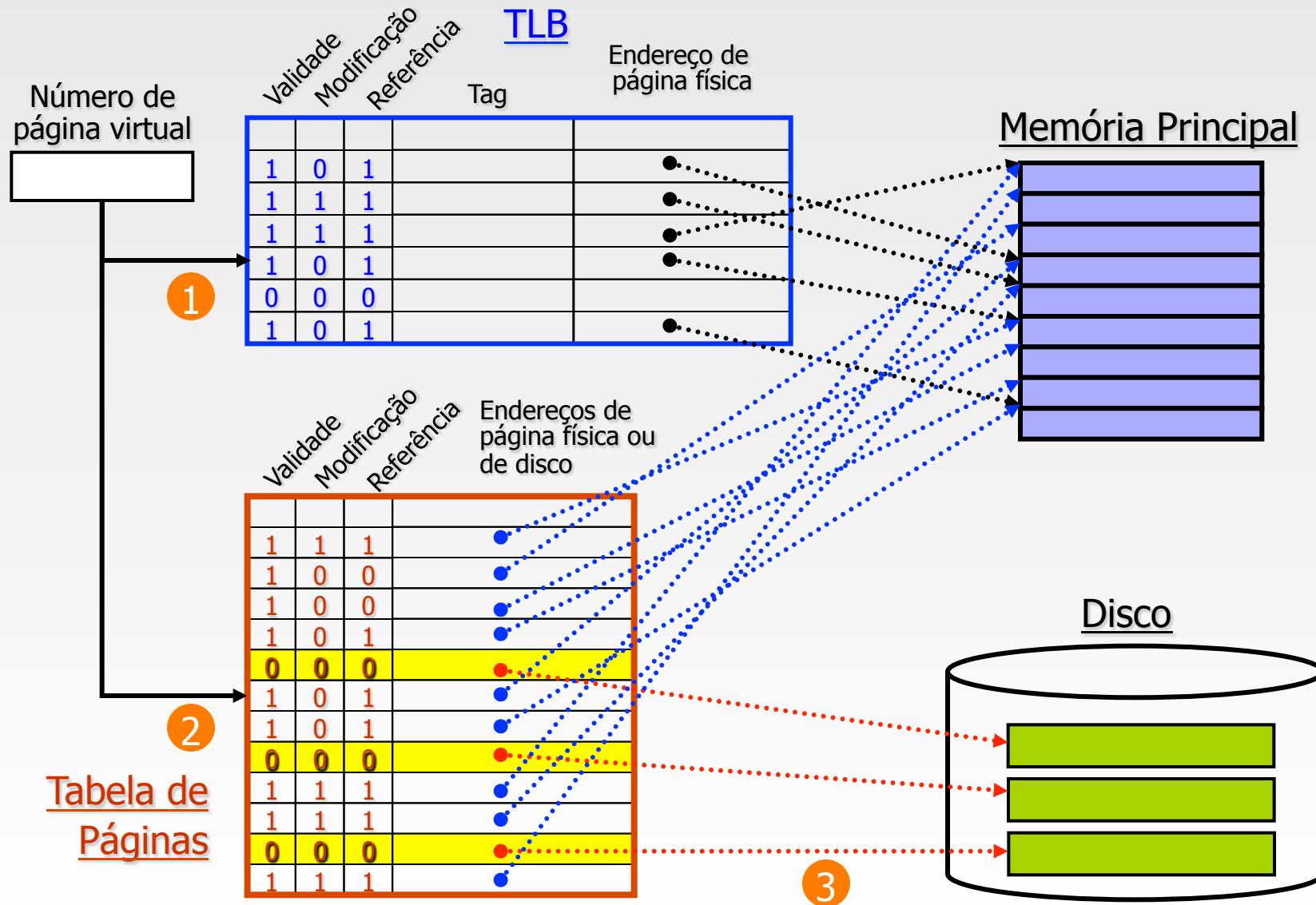
Tradução de endereços
com TLB

TLB – Translation Lookaside Buffer

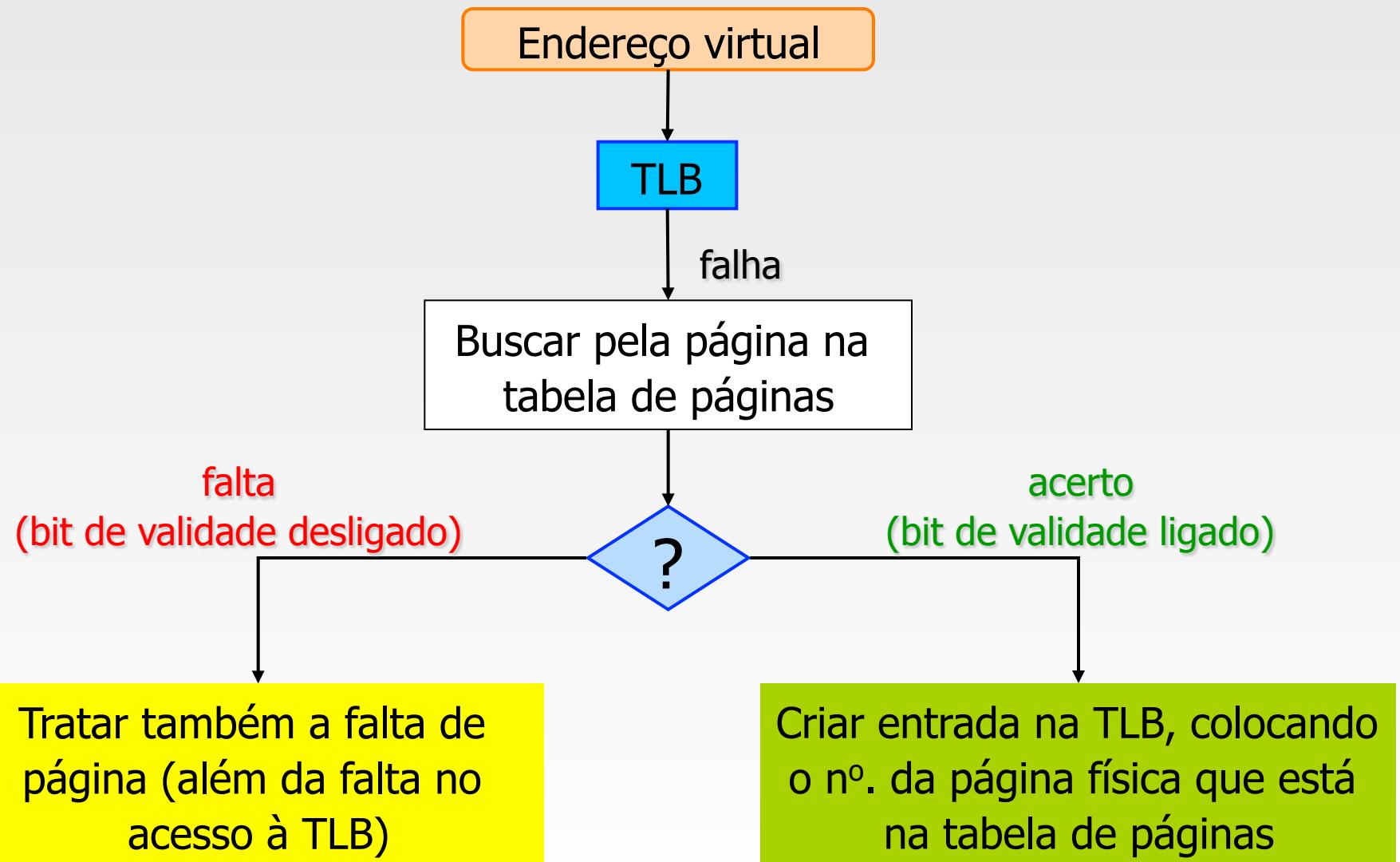
- ❖ Cada entrada da TLB contém:



TLB – Translation Lookaside Buffer



Tratando Faltas de Página e Falhas na TLB



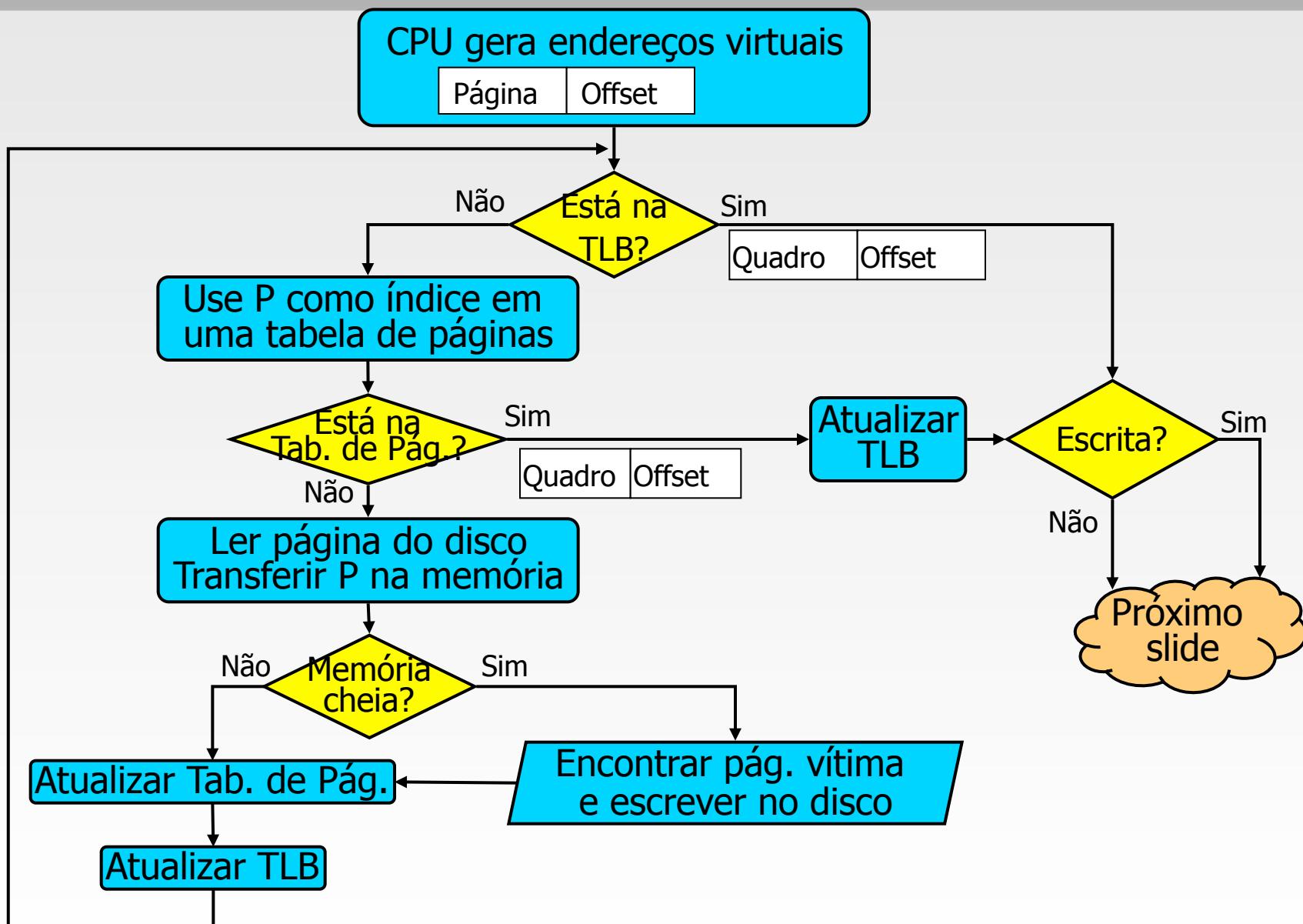
Tratando Faltas de Página e Falhas na TLB

- ❖ Falha na TLB pode ser tratada por SW ou por HW:
 - ▶ Requer uma curta seqüência de operações para copiar uma entrada válida da tabela de páginas da Mem. Princ. para a TLB
- ❖ O tratamento de **faltas de página** ou **falhas na TLB** requer uso do mecanismo de exceção para:
 - ▶ **Interromper** o processo ativo
 - ▶ **Transferir** o controle para o SO
 - ▶ **Retomar** a execução do processo interrompido
- ❖ Falta de página geralmente é reconhecida durante o ciclo de clock usado para **acessar a memória**
 - ▶ PC é salvo no registrador especial **EPC**, a fim de a execução ser retomada a partir da instrução seguinte

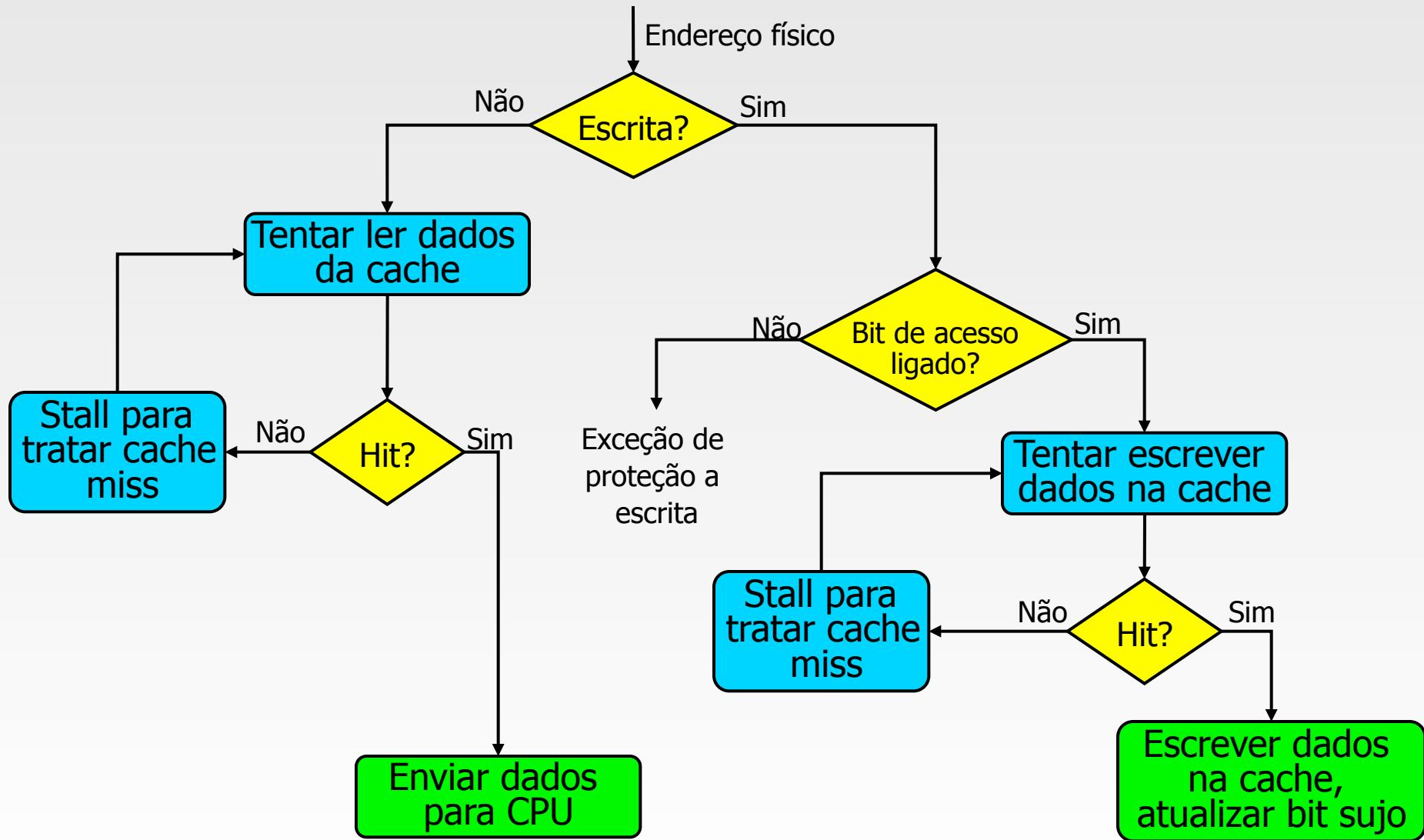
Integração da Memória Virtual, TLBs e Caches

- ❖ Dados **não podem estar na memória cache** a menos que estejam **presentes na memória principal**
- ❖ Quando o SO decide **migrar** uma página pouco usada para o disco, **deve garantir:**
 - ▶ **Remoção** do conteúdo de qualquer página da cache
 - ▶ **Modificação** das **tabelas de páginas** e da **TLB** de modo que uma tentativa de acessar dados da página antiga gere uma **falta de página**

Integração da Memória Virtual, TLBs e Caches



Integração da Memória Virtual, TLBs e Caches

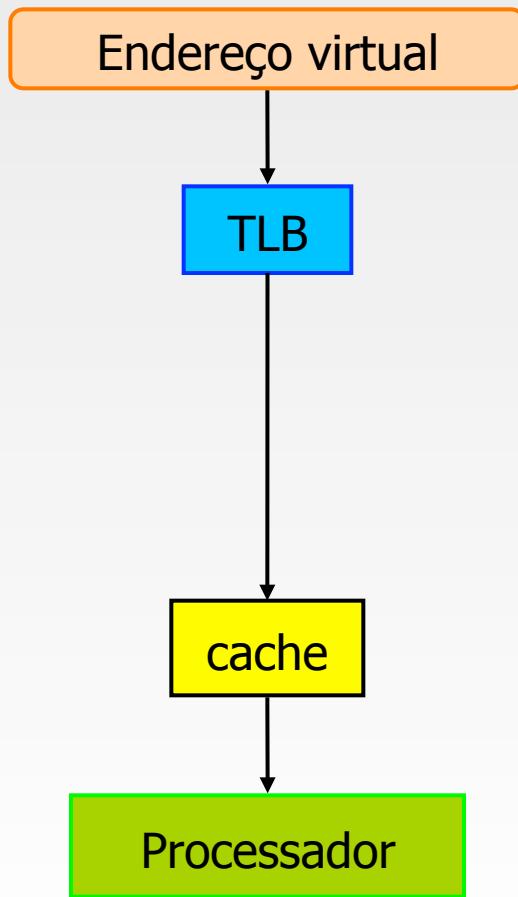


Integração da Memória Virtual, TLBs e Caches

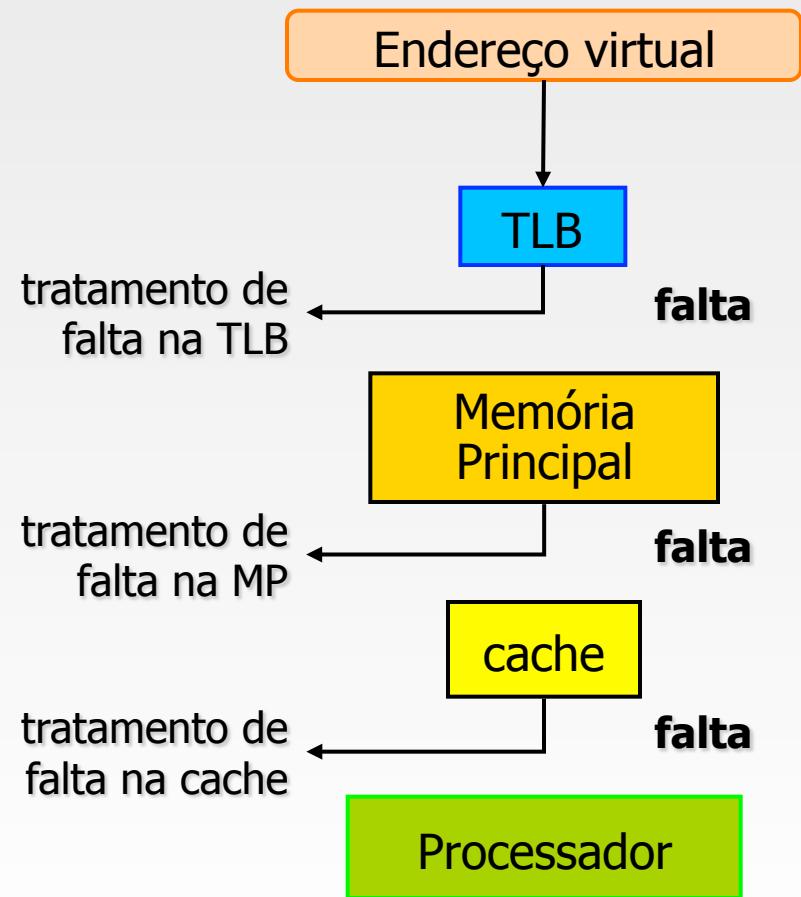
Cache	TLB	Tabela de páginas	Tradução possível? Se sim, em que circunstâncias?
falta	acerto	acerto	Possível, apesar de a tabela de páginas nunca ser verificada no caso de acerto no acesso à TLB
acerto	falta	acerto	Faltas no acesso à TLB, mas a entrada encontra-se na tabela de páginas; recuperadas as informações relativas ao mapeamento da tabela de páginas, a informação requisitada está na cache
falta	falta	acerto	Faltas no acesso à TLB, mas a entrada encontra-se na tabela de páginas; recuperadas as informações relativas ao mapeamento da tabela de páginas, a informação requisitada não está na cache
falta	falta	falta	Falta no acesso à TLB seguida de uma falta de página; após recuperar as informações de mapeamento, deve ser necessariamente gerada uma falta no acesso à cache
falta	acerto	falta	Impossível: não pode haver tradução na TLB se a página não está presente na memória
acerto	acerto	falta	Impossível: não pode haver tradução na TLB se a página não está presente na memória
acerto	falta	falta	Impossível: não é permitido que a informação esteja na cache se a página não está na memória principal

Integração da Memória Virtual, TLBs e Caches

Condição ideal



Pior caso



Proteção com Memória Virtual

❖ Função de um sistema de memória virtual:

Permitir que uma única memória principal seja compartilhada por vários processos, oferecendo proteção de memória entre eles e o SO

❖ Mecanismo de proteção precisa garantir que:

- Um processo “rebelde” não possa escrever no espaço de endereçamento de outro processo de usuário ou no espaço do SO
- Um determinado processo não leia dados de outro processo

Proteção com Memória Virtual

- ❖ Um processo deve operar corretamente, esteja em:
 - ▶ Execução **contínua**, ou
 - ▶ Interrupção **repetida** e **alternante** com outros processos
- ❖ A responsabilidade pelo comportamento correto dos processos é compartilhada entre:
 - ▶ Arquitetura – deve assegurar que o **estado da CPU** seja **salvo** e **restaurado**, ao rodar um determinado processo
 - ▶ Sistema Operacional – processos **não podem interferir** uns com os outros

Proteção com Memória Virtual

- ❖ Cada processo tem, ao mesmo tempo, na memória principal:
 - ▶ Espaço de **endereçamento virtual**
 - ▶ Estado da CPU
 - ▶ Espaço de **instruções**
 - ▶ Espaço de **dados**
- ❖ O SO deve manter as tabelas de páginas **organizadas** de modo que:
 - ▶ Páginas virtuais independentes sejam mapeadas para **páginas físicas disjuntas**
 - ▶ Assim, um processo **não será capaz** de acessar as informações de outro

Proteção com Memória Virtual

Um processo de usuário **não é capaz de modificar o mapeamento da tabela de páginas**



O **SO proíbe** o processo de usuário de modificar sua própria tabela de páginas. Mas o **SO precisa poder modificar** qualquer das tabelas de páginas



Por isso, as tabelas de páginas são colocadas no espaço de endereçamento do SO

Proteção com Memória Virtual

- ❖ Suporte de hardware para que o SO implemente proteção no sistema de memória virtual:
 1. Oferecer pelo menos dois modos de execução de processos:
 - usuário (processo de usuário)
 - supervisor/kernel (processo do SO)
 2. Fornecer uma parte do estado que um processo usuário possa ler mas não possa escrever (bit de modo usuário/supervisor + TLB + um ponteiro para a tabela de páginas)
 3. Fornecer mecanismos para trocar de modo:
 - usuário → supervisor: feita via exceção de chamada de sistema, implementada por uma instrução especial (*syscall* no MIPS)
 - supervisor → usuário: executar uma instrução retorno de exceção

Proteção com Memória Virtual

:: Compartilhando informações entre processos

- ❖ Processos podem precisar compartilhar informações entre si para leitura, mas não para escrita
- ❖ Exemplos:
 - ▶ Múltiplas instâncias de uma mesmo programa em execução
 - ▶ Múltiplas instâncias de uma mesma biblioteca em uso por vários programas
- ❖ Portanto, um bit de acesso à escrita pode ser incluído na tabela de páginas a fim de restringir o compartilhamento apenas à leitura
- ❖ Como o restante da tabela de páginas, esse bit de acesso à escrita só pode ser modificado pelo SO

Proteção com Memória Virtual

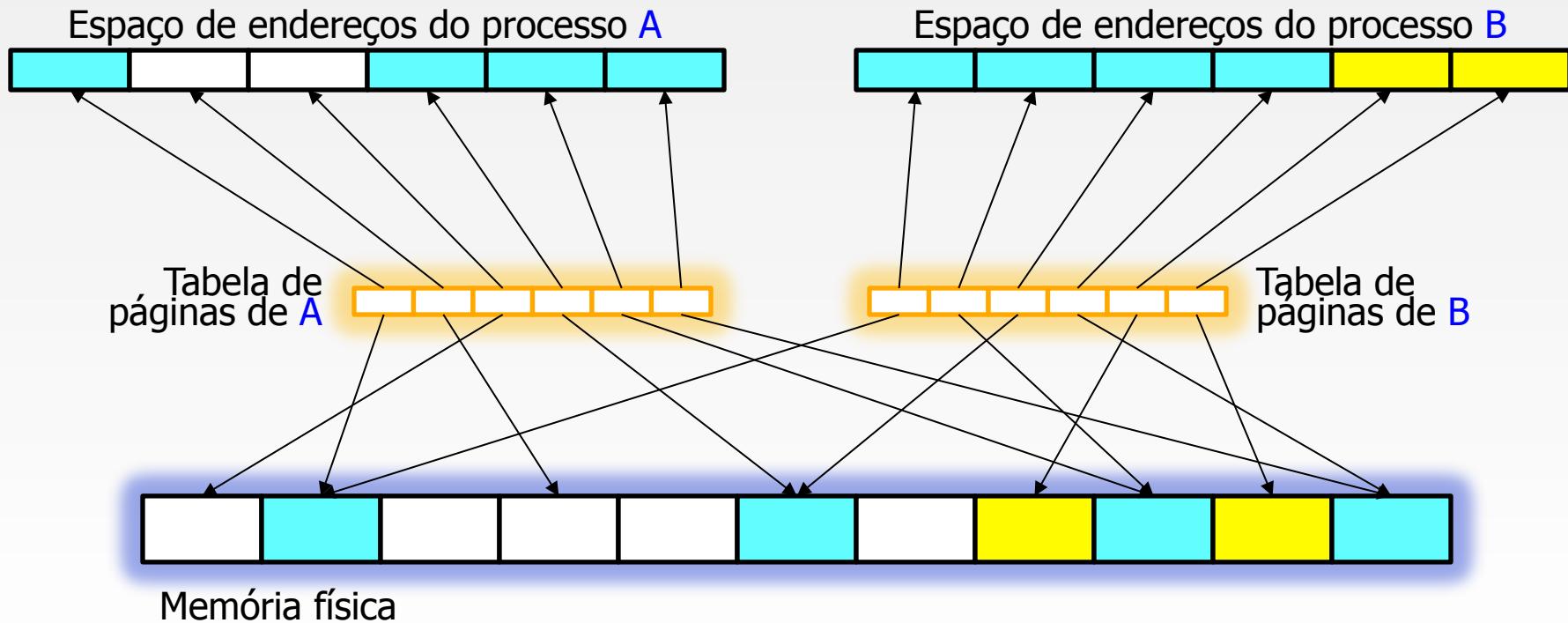
:: Compartilhando informações entre processos

- ❖ Para permitir que um processo A leia uma página pertencente ao processo B:
 - ▶ B solicita ao SO a **criação de uma entrada** na tabela de páginas para uma página virtual no espaço de **endereçamento** de A que **aponte para uma página física** que B deseja **compartilhar**
 - ▶ SO pode usar o bit de escrita para **evitar que A escreva** nas informações acessadas, se B assim o desejar
 - ▶ Bits que determinam os direitos de acesso a uma página precisam ser incluídos na **tabela de páginas** e na **TLB**

Proteção com Memória Virtual

:: Compartilhando informações entre processos

- ❖ Processos A e B compartilham várias páginas
 - Suas tabelas de páginas informam onde as páginas virtuais estão alocadas na memória física



Resumo

❖ Memoria virtual

- ▶ Conceitos
- ▶ Tradução do endereços
- ▶ Tabelas de páginas
- ▶ Translation lookaside Buffer (TLB)
- ▶ Proteção