Sistemas Operacionais

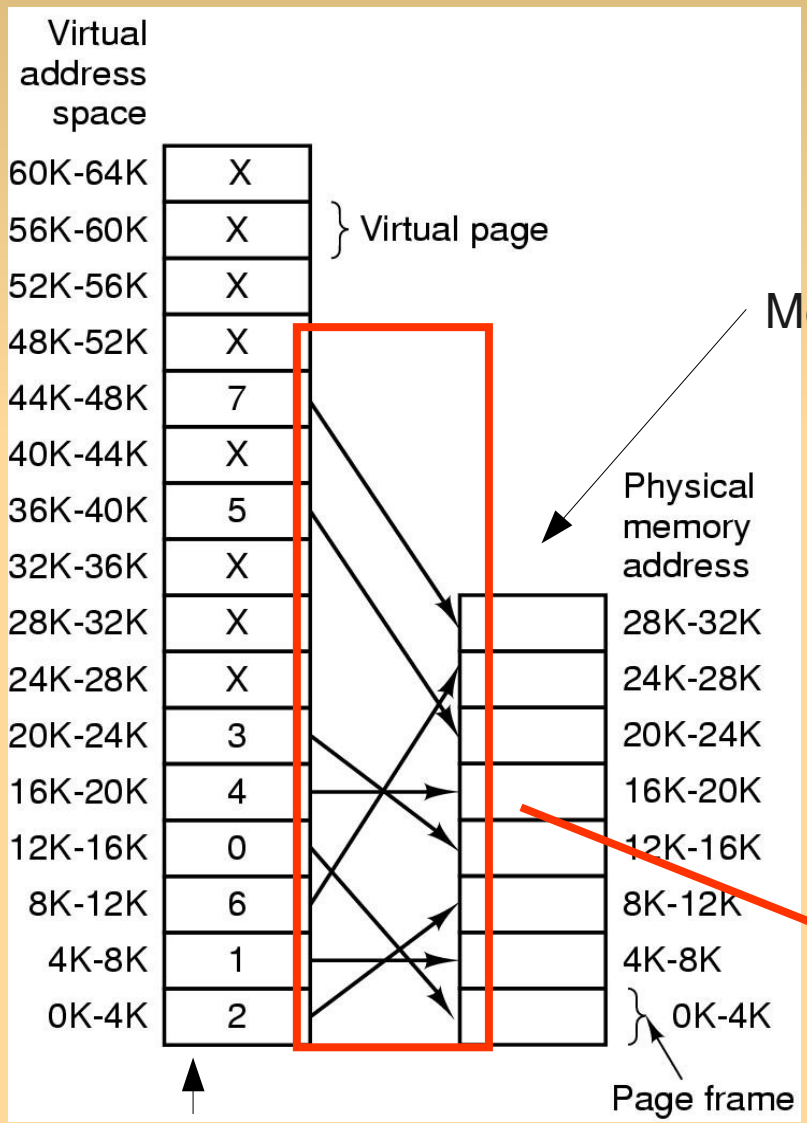
Gerenciamento de Memória Virtual
Paginação

Norton Trevisan Roman
Marcelo Morandini
Jó Ueyama

Apostila baseada nos trabalhos de Kalinka Castelo Branco,
Antônio Carlos Sementille, Luciana A. F. Martimiano e nas transparências
fornecidas no site de compra do livro "Sistemas Operacionais Modernos"

Memória Virtual

- Página virtual **mapeada** para página real;

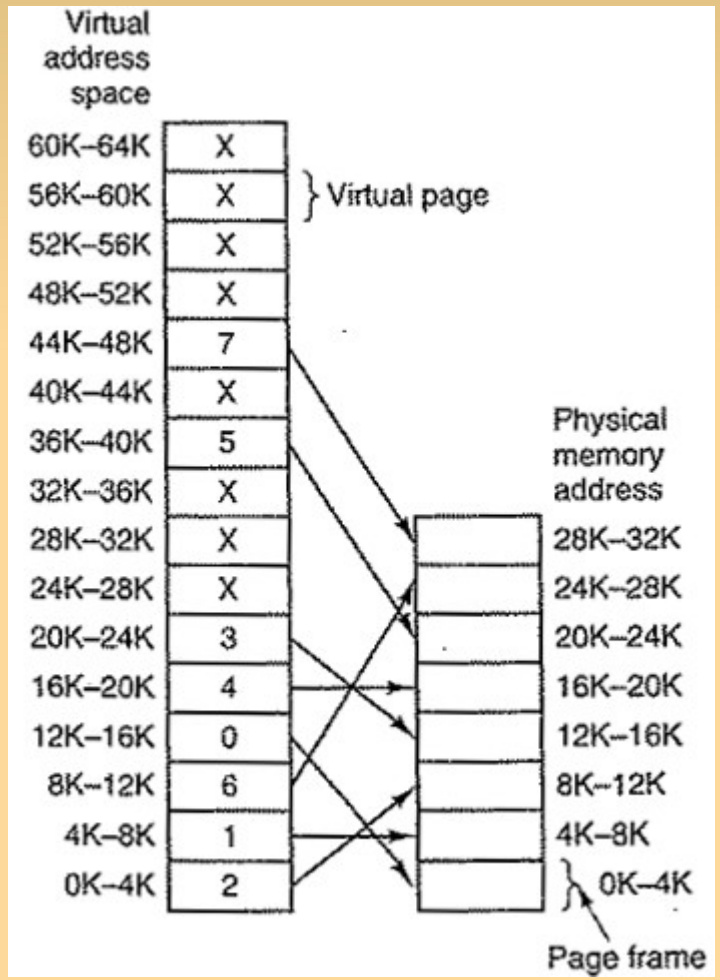


- Ex:
  - Computador que gera 64K de endereços virtuais
    - Tem apenas 32K de endereços físicos
    - Programas de 64K podem ser escritos, mas não carregados inteiramente na memória

- MMU realiza o mapeamento

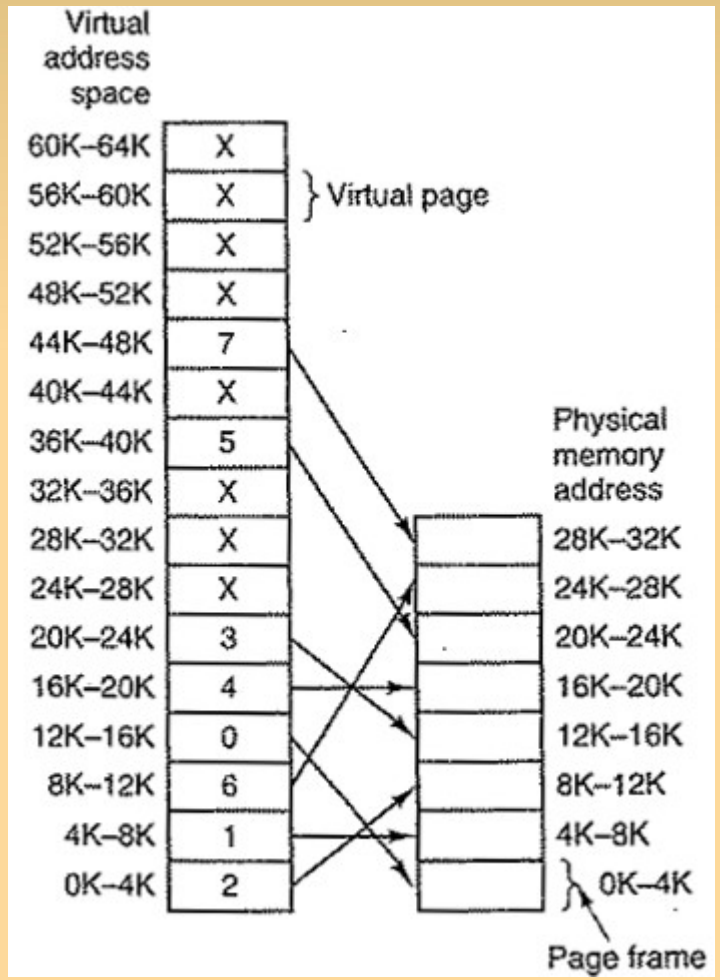
Memória Virtual

- Divide-se o espaço de endereçamento virtual em unidades de tamanho fixo – as páginas
  - Nesse caso, Páginas de 4Kb
    - 4096 bytes/endereços (0-4095)
  - As unidades correspondentes na memória física são as page frames
  - Geralmente, pages e page frames têm o mesmo tamanho



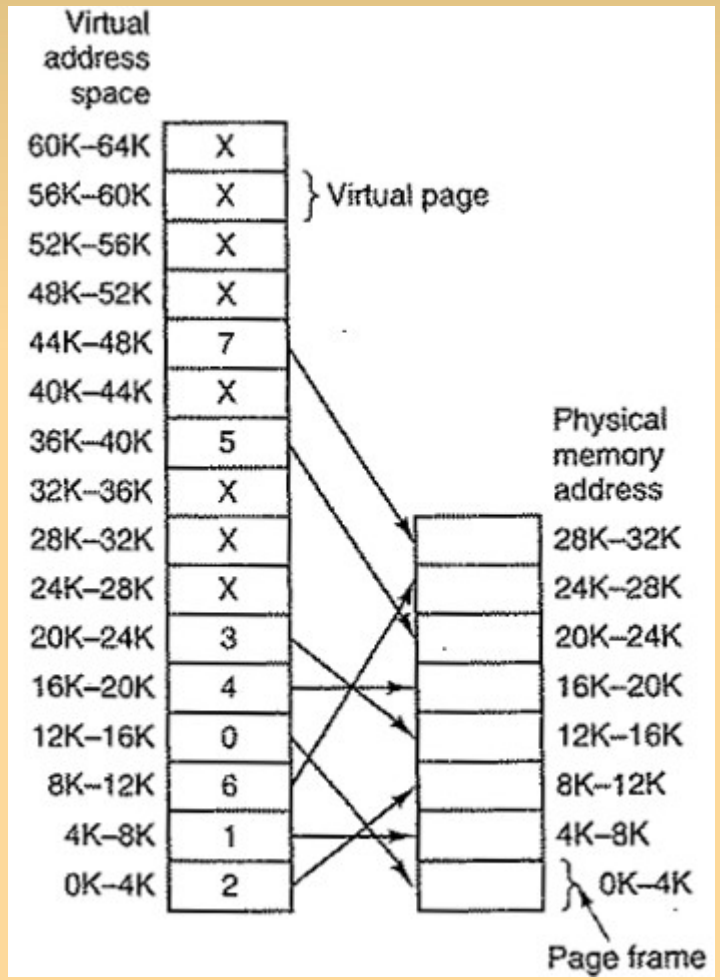
Paginação
Volta ao Exemplo

- Embora tenha 32KB, o sistema age como se tivesse 64KB
- MOV REG,5
  - A MMU identifica que é a primeira página (5B acima da sua base → 0)
  - Ela está mapeada à terceira frame, que começa em 8k = 8192
    - O endereço enviado ao barramento é 5 + 8192 = 8197



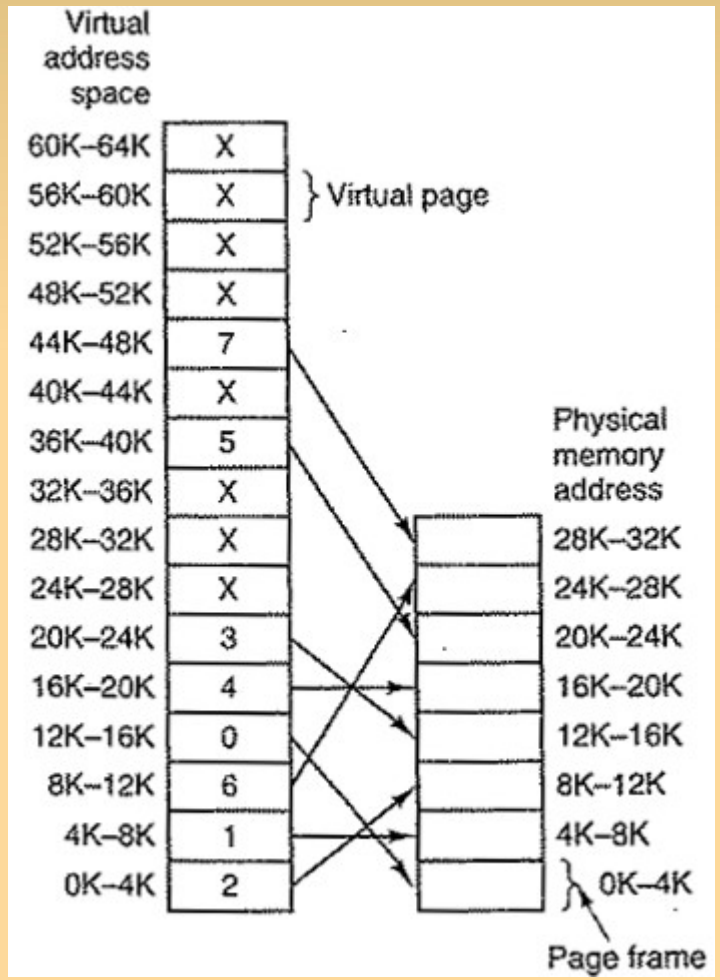
Paginação

- Como sabemos que páginas estão na memória efetivamente?



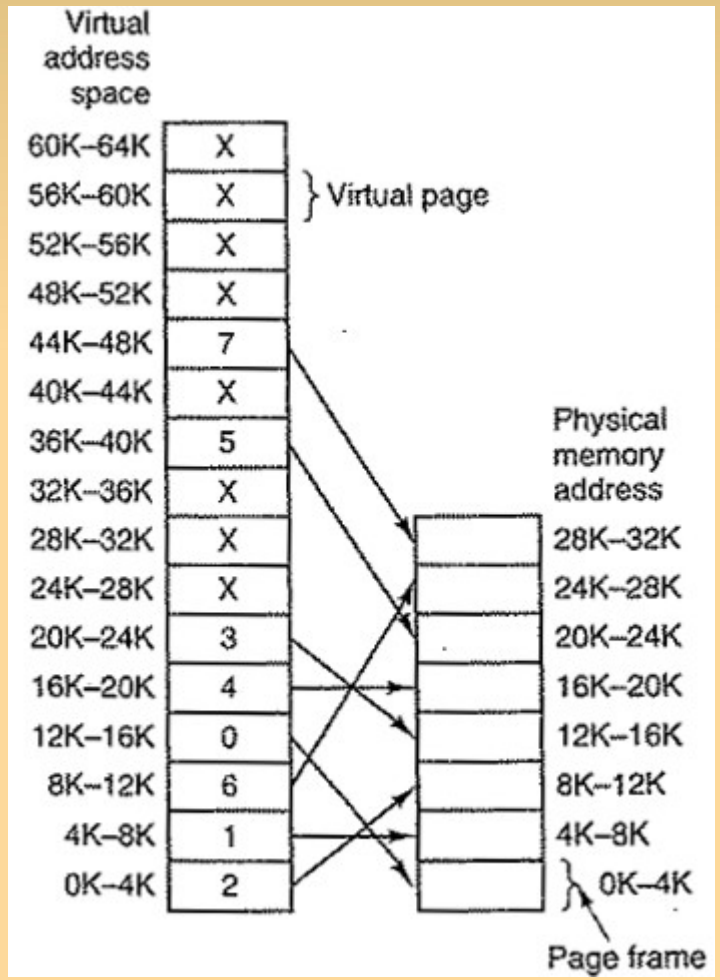
Paginação

- Como sabemos que páginas estão na memória efetivamente?
  - Se temos apenas 8 frames, somente 8 páginas (das 16) estão mapeadas
  - Solução:
    - Bit de presente/ausente
    - Identifica que páginas estão fisicamente presentes na memória



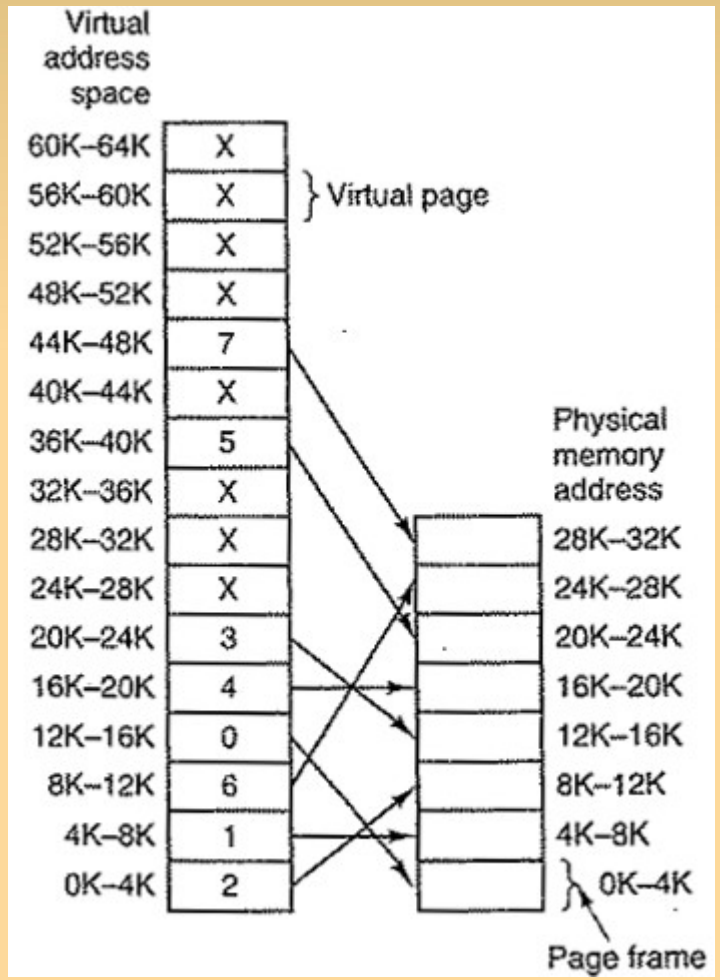
Paginação

- E se um programa referenciar um endereço não mapeado?
  - Ex: MOV REG, 32780
    - Byte 12 da página 8



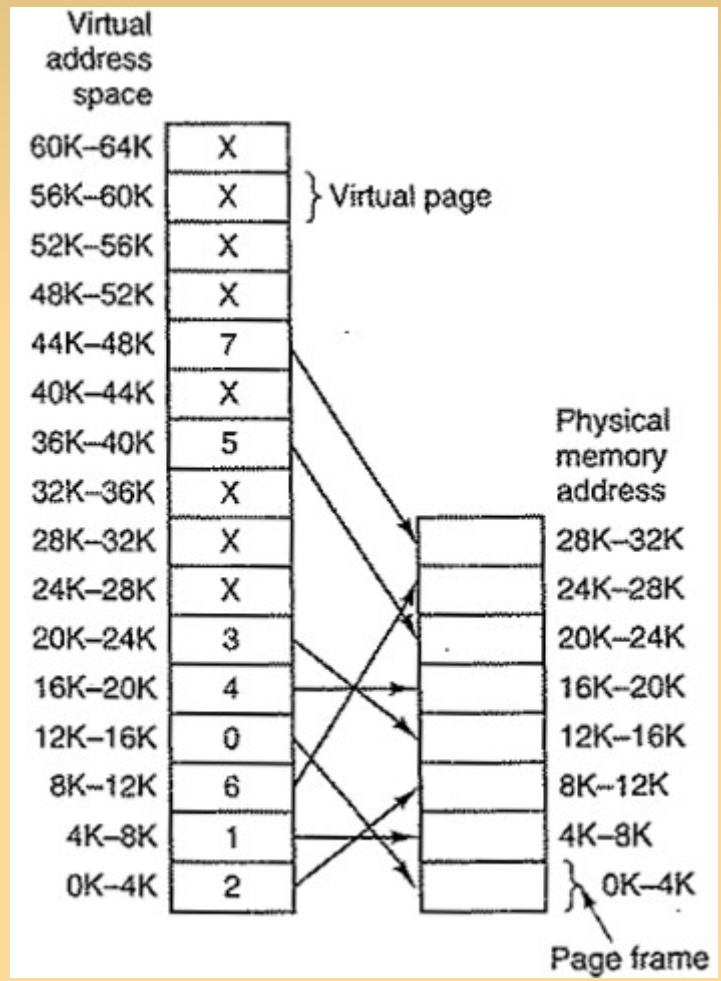
Paginação

- E se um programa referenciar um endereço não mapeado?
  - Ex: MOV REG, 32780
    - Byte 12 da página 8
  - A MMU verifica que a página não está mapeada
    - Força o desvio da CPU para o S.O., via interrupção (trap) – page fault



Paginação

- E se um programa referenciar um endereço não mapeado?
  - O S.O. toma uma página pouco usada
    - Escreve seu conteúdo no disco
  - Carrega a página recém referenciada na moldura (frame) recém liberada
    - Muda o mapa (veremos mais adiante)
    - Reinicia a instrução aprisionada no trap (que causou a interrupção)



Memória Virtual e Paginação

- Então...
  - Memória Virtual pode ser implementada quebrando-se o espaço de endereçamento virtual em páginas
  - E mapeando cada página a alguma moldura de página na memória física
  - Ou mantendo-a temporariamente não mapeada

Paginação

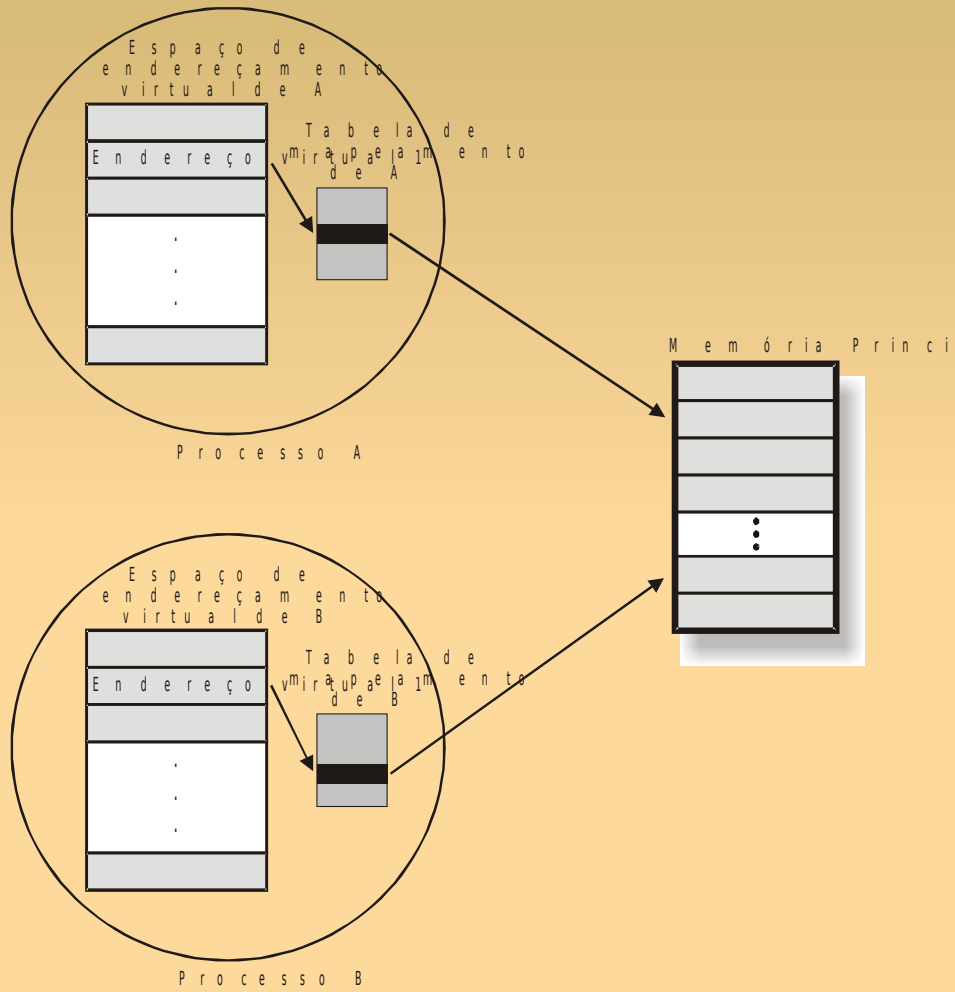
- Memória Principal e Memória Secundária são organizadas em páginas de mesmo tamanho;
  - Página é a unidade básica para transferência de informação;
- E como se dá o mapeamento?

Paginação

- Memória Principal e Memória Secundária são organizadas em páginas de mesmo tamanho;
  - Página é a unidade básica para transferência de informação;
- E como se dá o mapeamento?
  - Tabela de páginas

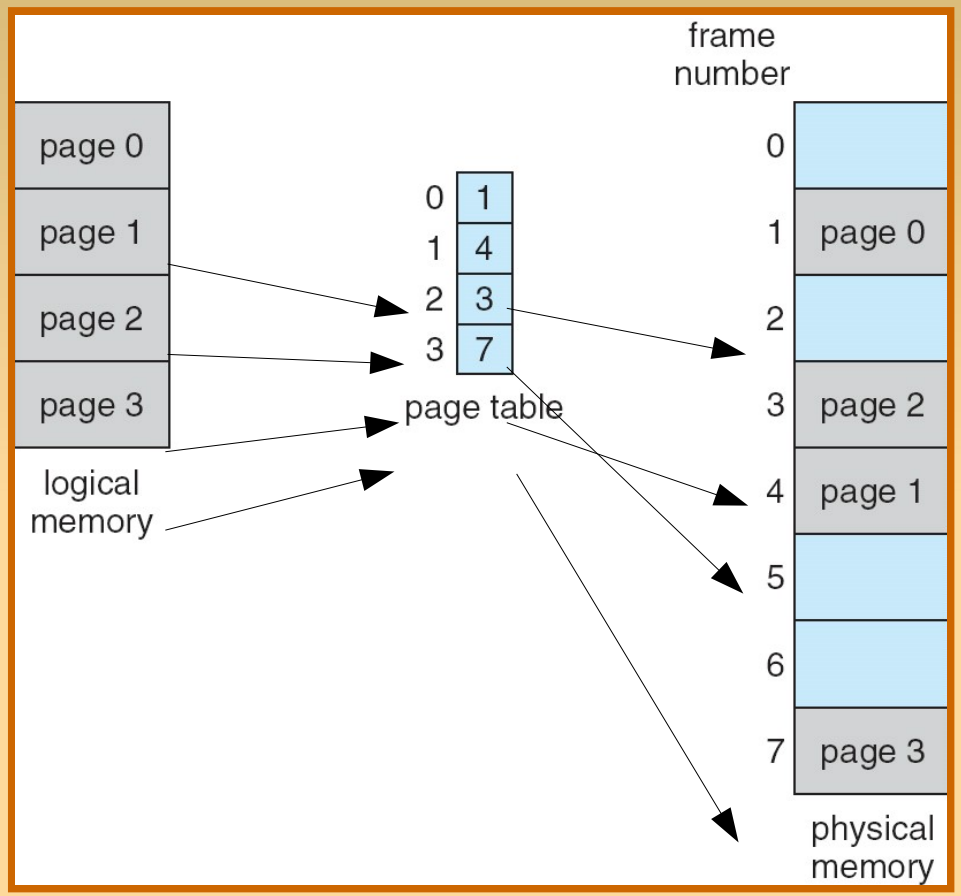
Paginação – Tabela de Páginas

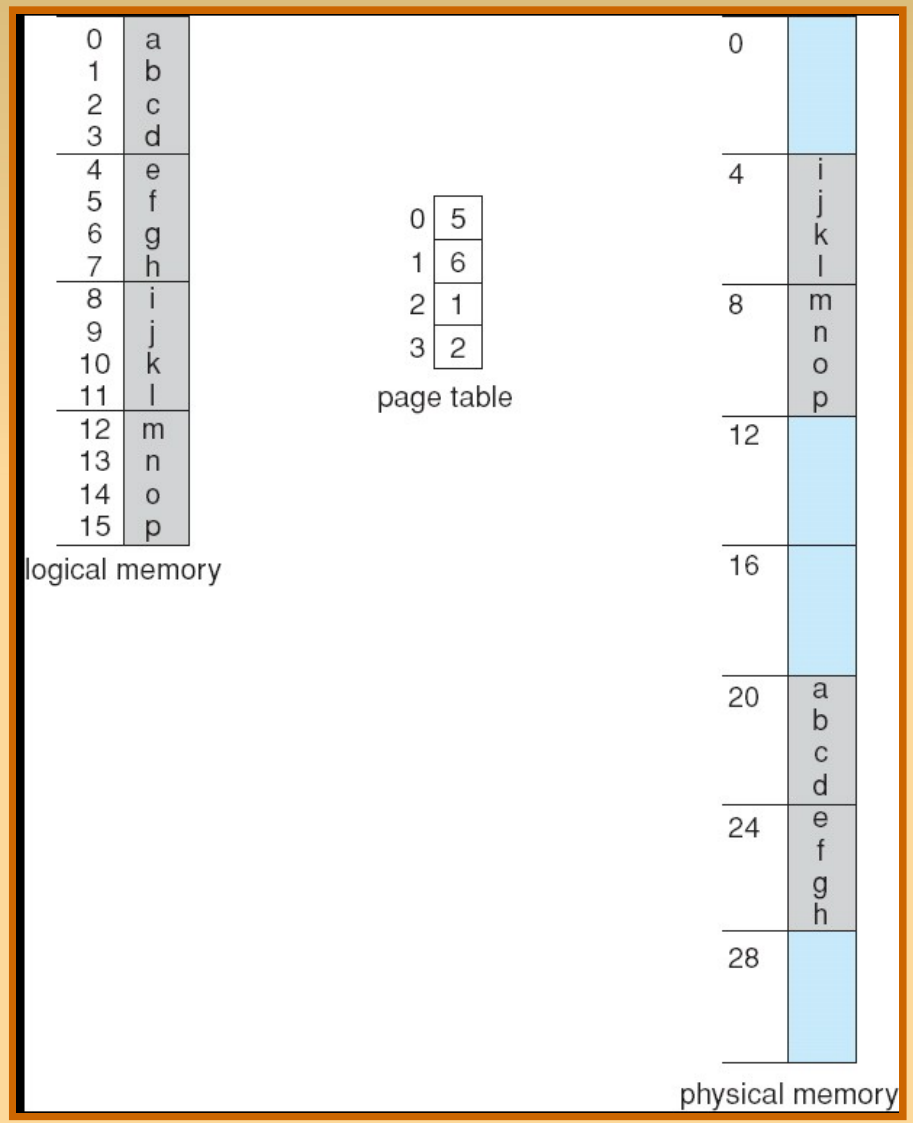
- Tabela de páginas:
  - Responsável por armazenar informações sobre as páginas virtuais:
    - argumento de entrada → número da página virtual;
    - argumento de saída (resultado) → número da página real (ou moldura de página - page frame);



- Cada processo tem sua própria tabela
  - Cada um tem seu próprio espaço de endereçamento
  - Cada um acha que começa em uma mesma posição

Paginação – Tabela de Páginas





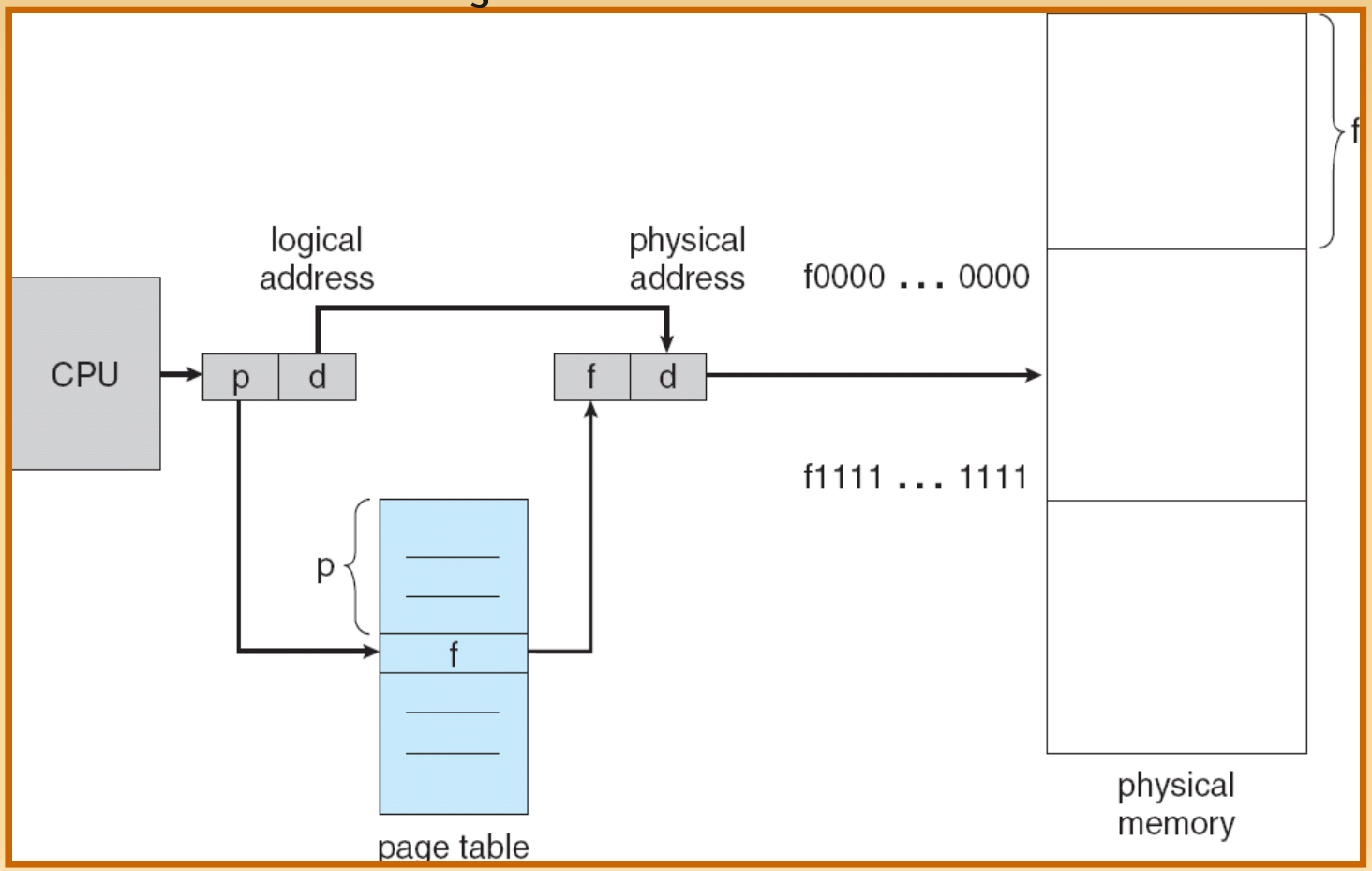
Embora mapeie páginas a molduras, consegue mapear todo o conteúdo dentro delas

Paginação – Tabela de Páginas

- E como buscar um endereço?
  - Busca seqüencial? Binária?
    - Qualquer que seja a alternativa, é lenta
  - Que fazer?

Paginação – Funcionamento da Tabela de Páginas na MMU

- E como buscar um endereço?
  - Ideal: usar parte do endereço virtual como índice na tabela, onde está o endereço-base da moldura correspondente na memória



Paginação – Funcionamento da Tabela de Páginas na MMU

- E como buscar um endereço?
  - Ideal: usar parte do endereço virtual como índice na tabela, onde está o endereço -base da moldura correspondente na memória
  - Como fazer isso?
    - Use tamanhos de página que sejam potências de 2
    - 4K = 4096 = 2^{12}
      - 0k a 4k → 000000000000000 a 001000000000000
      - 4k a 8k → 001000000000000 a 010000000000000
      - 8k a 12k → 010000000000000 a 011000000000000
      - 12k a 16k → 011000000000000 a 100000000000000
      - . . .

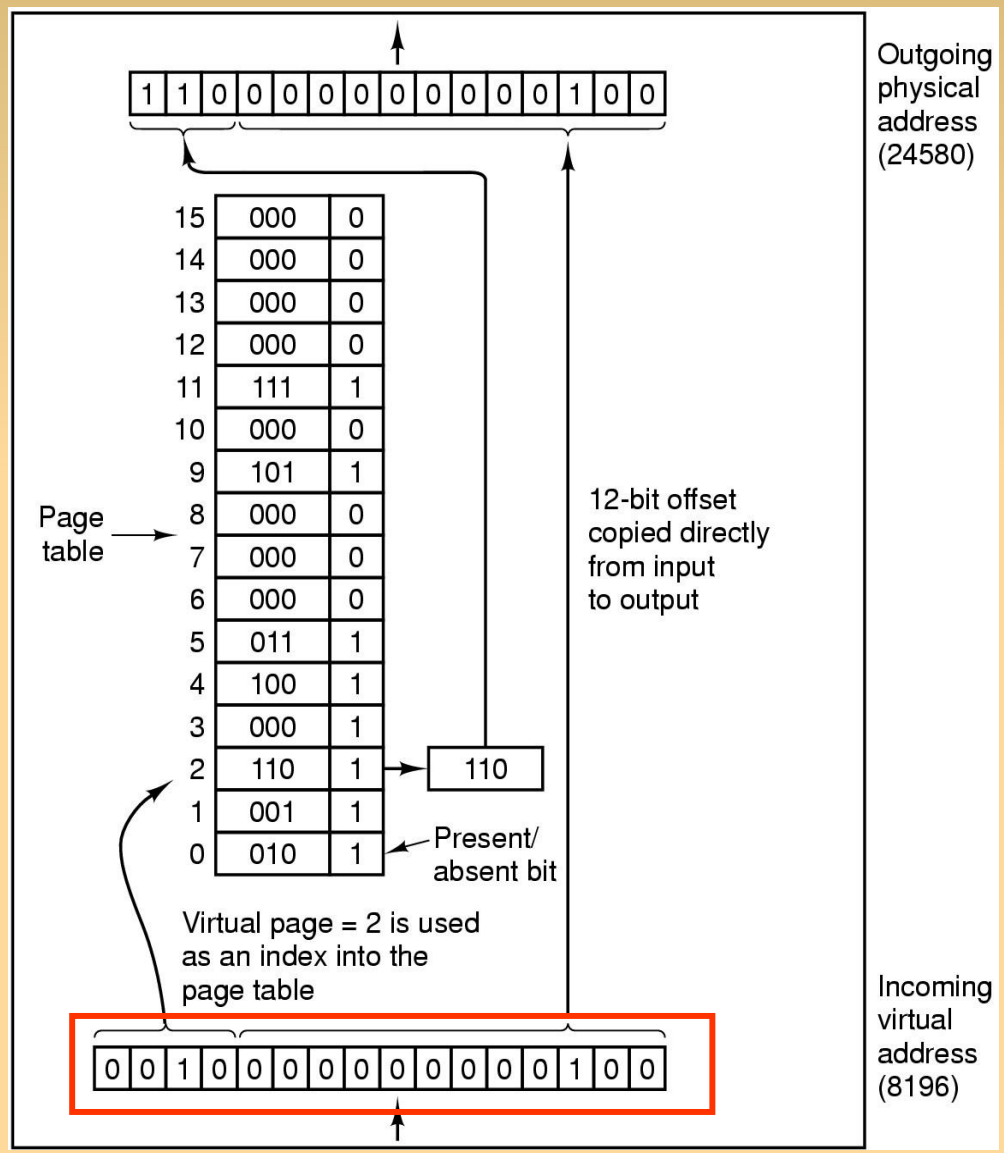
Paginação – Funcionamento da Tabela de Páginas na MMU

- E como buscar um endereço?
  - E endereços dentro da página?
    - Use os bits além do limite da página (em preto, no exemplo anterior)
    - Ex: 8196
      - Múltiplo de 4k mais próximo → 2 = 8192
      - 8196 = 8192 + 4
      - (8192) 001000000000000 +
      - (4) 000000000000100 =
      - (8196) 001000000000100
    - Parte em vermelho: endereço-base da página
    - Parte em preto: deslocamento (offset)

Paginação – Funcionamento da Tabela de Páginas na MMU

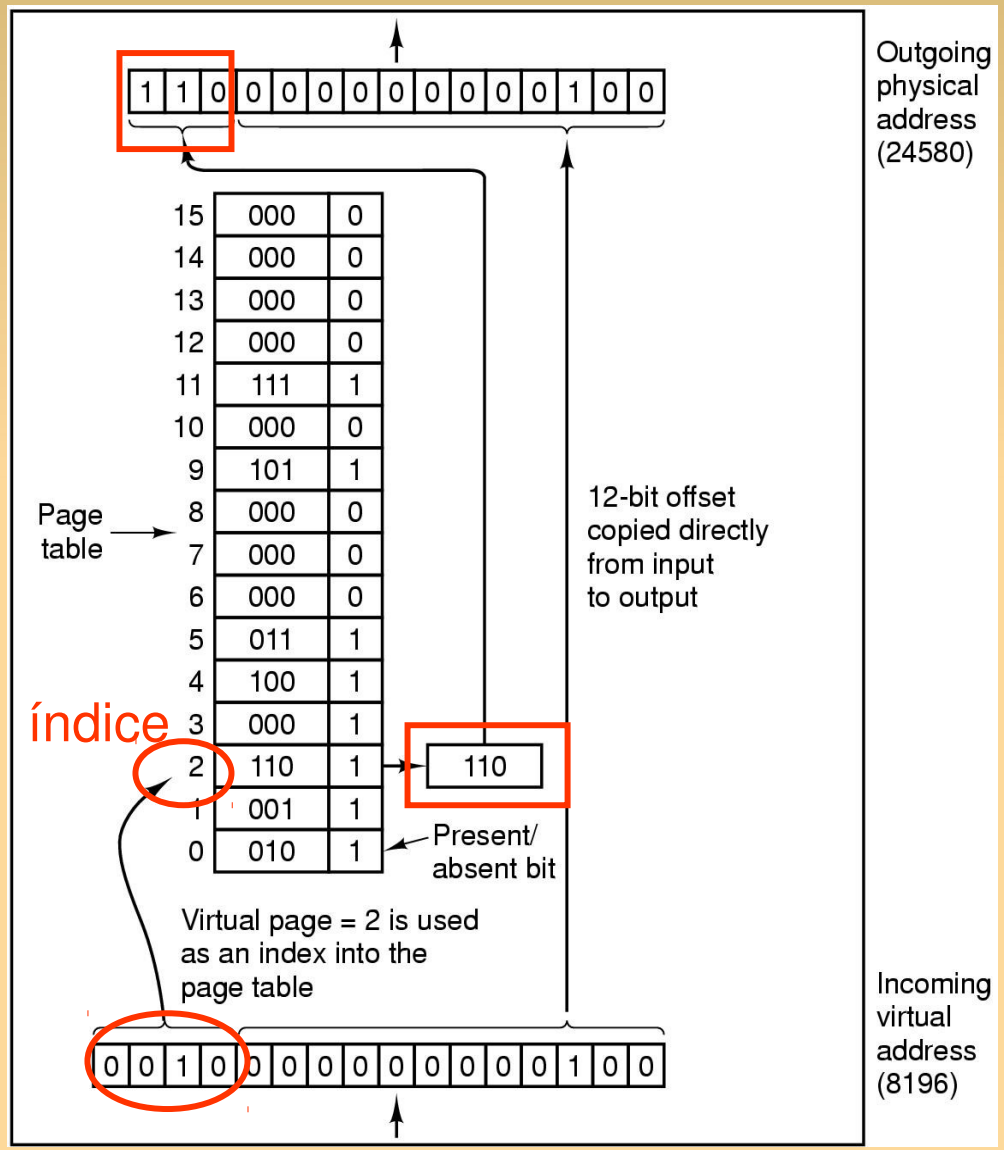
- E como buscar um endereço?
  - Parte em vermelho:
    - Endereço-base da página (montado zerando-se a parte em preto)
    - Varia, na hora de mapear páginas a suas respectivas molduras
  - Parte em preto:
    - Deslocamento (offset)
    - Não varia → um endereço que estava n bytes acima da base da página estará os mesmos n bytes acima da base da moldura

Paginação – Funcionamento da Tabela de Páginas na MMU



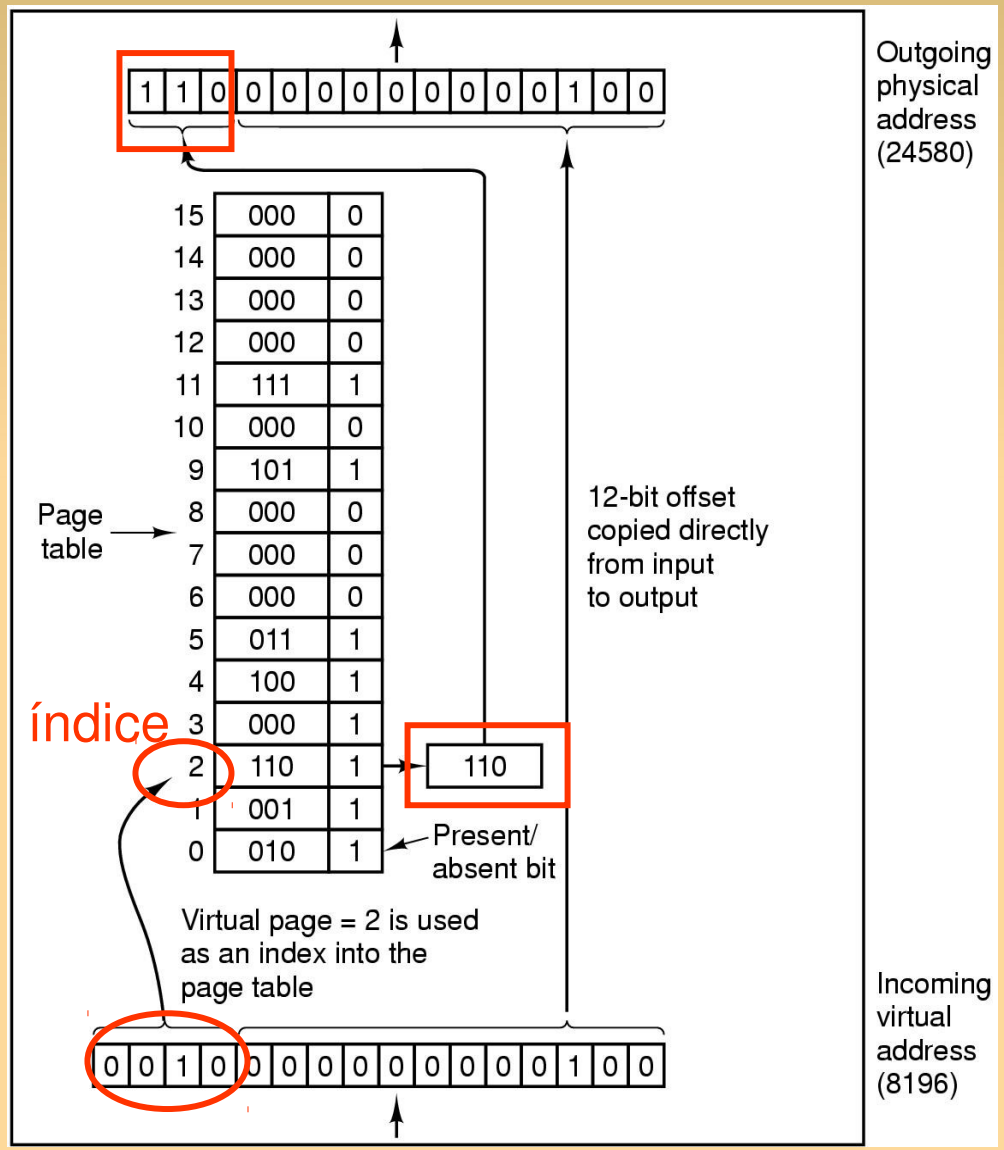
- Ex: Endereço 8196
  - 0010000000000100
  - MMU com 16 páginas de 4Kb
    - Endereço virtual de 16 bits
    - A tabela tem 16 entradas (0000 a 1111)
  - Hardware com 8 frames
    - Endereço físico de 15 bits

Paginação – Funcionamento da Tabela de Páginas na MMU



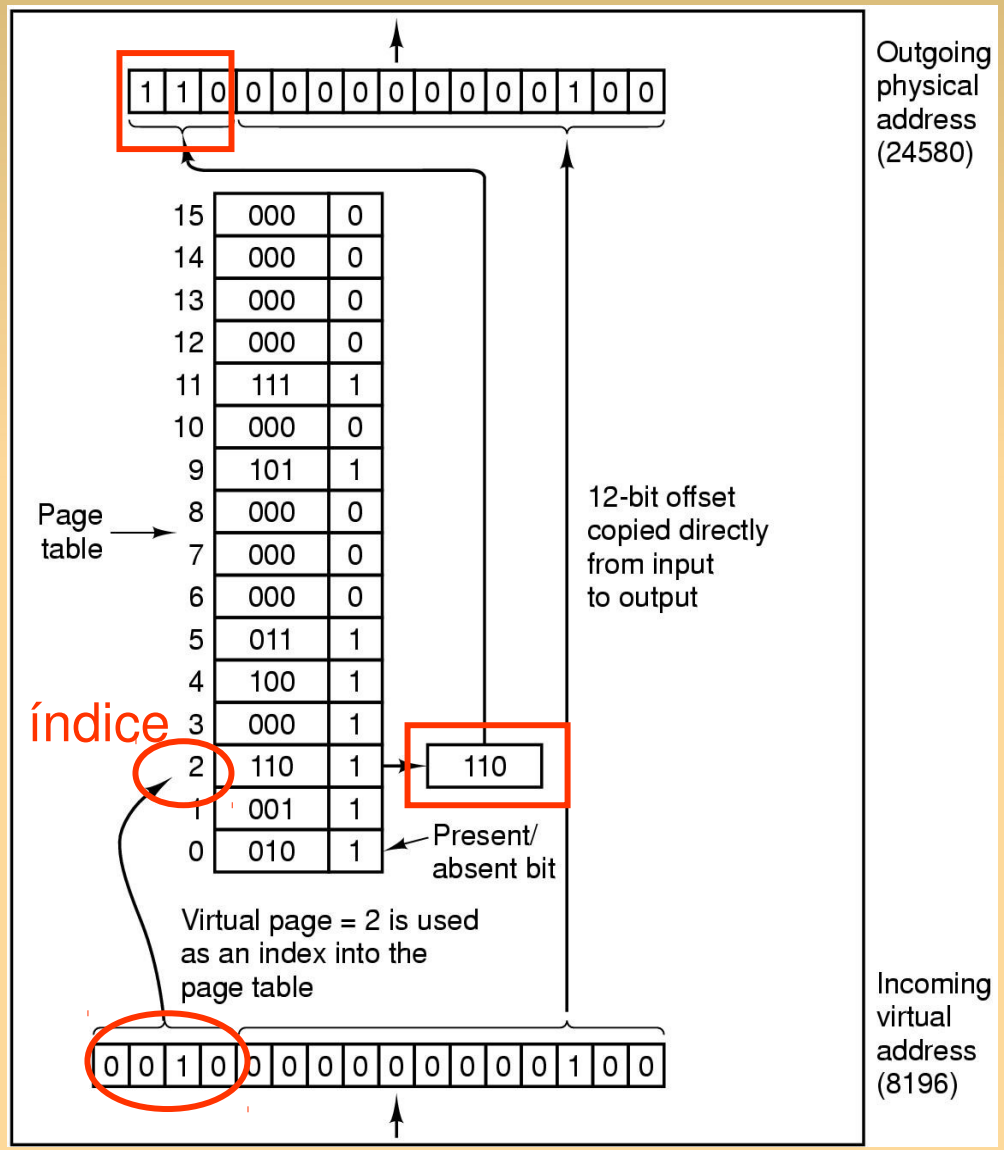
- Usamos os 4 bits mais altos do endereço virtual como índice na tabela
  - Se página estiver na RAM
    - (Bit presente/ausente = 1)
    - O endereço físico é montado

Paginação – Funcionamento da Tabela de Páginas na MMU



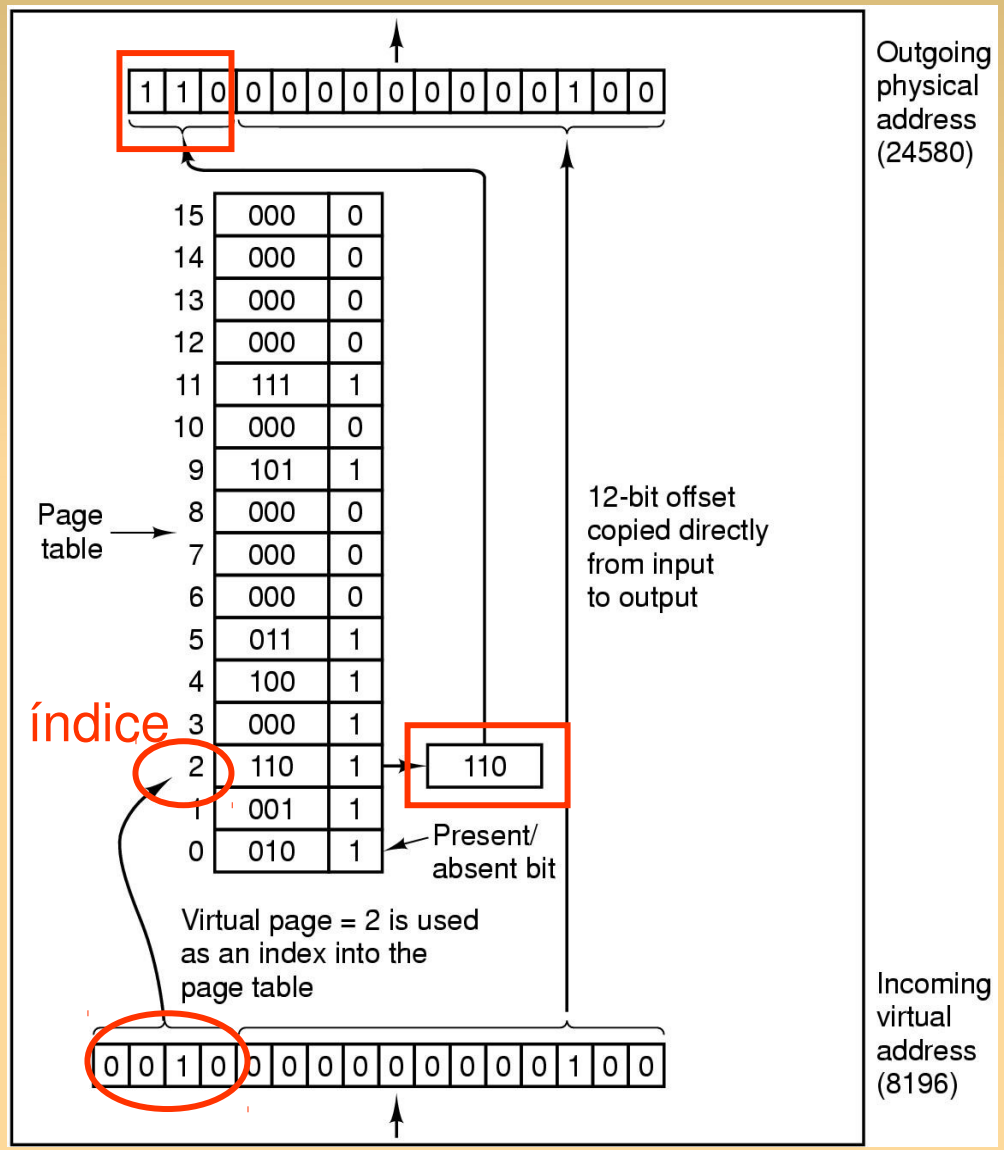
- Montando o endereço físico
  - O nº da página física (110) é copiado para os três bits mais significativos do endereço de saída (real), juntamente com o deslocamento (sem alteração)

Paginação – Funcionamento da Tabela de Páginas na MMU



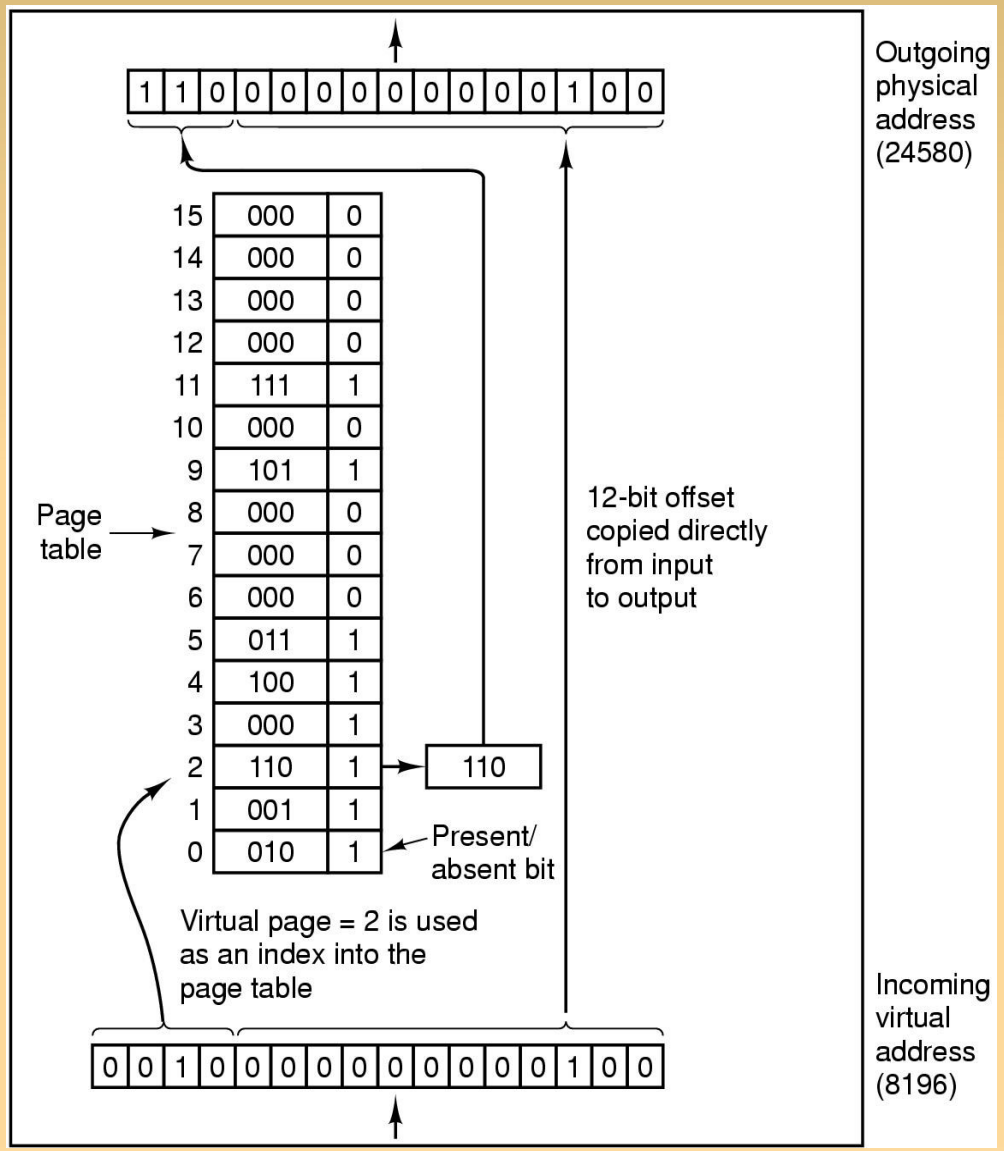
- Montando o endereço físico
  - O registrador de saída envia então esse endereço à memória, via barramento

Paginação – Funcionamento da Tabela de Páginas na MMU



- Usamos os 4 bits mais altos do endereço virtual como índice na tabela
  - Se página não estiver na RAM
    - (Bit presente/ausente = 0)
    - Executa uma trap → Desvia ao S.O.

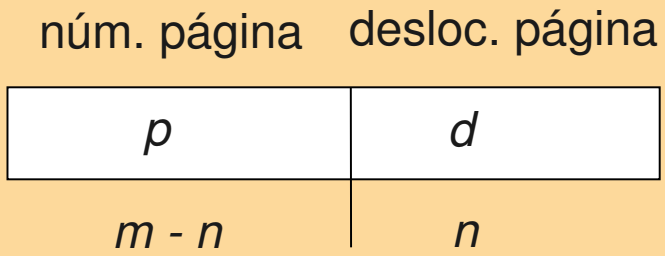
Paginação
Funcionamento da MMU



- Deslocamento:
  - O quanto acima da base da página está
- Ex:
  - Endereço 8296:
    - 1000001101000
  - 8296 = 8192 + 104:
    - 010000000000000 + 000000001101000 = 010000001101000

Esquema de Tradução de Endereço

- O endereço gerado pela CPU é dividido em:
  - Número de página (p) – usado como um índice para uma tabela de página que contém endereço de base de cada página na memória física
  - Deslocamento de página (d) – combinado com endereço de base para definir o endereço de memória físico que é enviado à unidade de memória



  - Para determinado espaço de endereço lógico 2^m e tamanho de página 2^n

Paginação

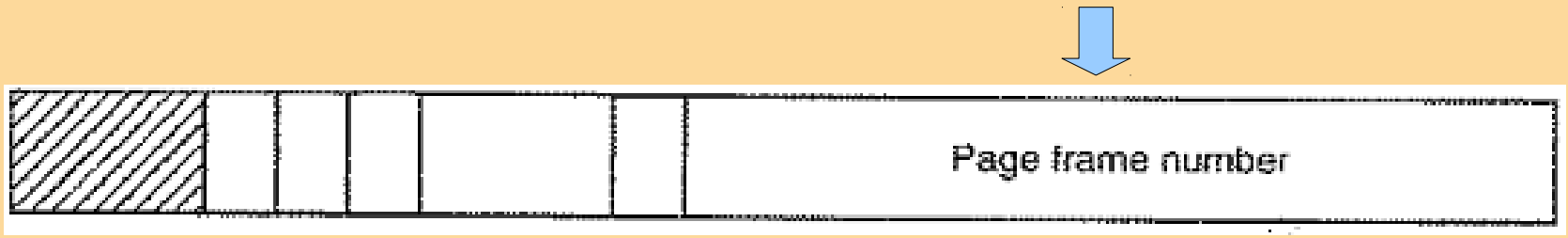
- Espaço de endereço lógico de um processo pode ser não contíguo
  - Processo recebe memória física sempre que houver memória disponível
- Implementação de paginação:
  - Divida a memória física em blocos de tamanho fixo, denominados molduras
  - Divida a memória lógica em blocos do mesmo tamanho, denominados páginas
  - Acompanhe todos os quadros livres
  - Configure uma tabela de página para traduzir endereços lógicos para físicos

Paginação

- Problemas:
  - Fragmentação interna;
  - Definição do tamanho das páginas;
    - Geralmente a MMU que define e não o SO;
    - Páginas maiores: leitura mais eficiente, tabela menor, mas maior fragmentação interna;
    - Páginas menores: leitura menos eficiente, tabela maior, mas menor fragmentação interna;
  - Problema:
    - Geralmente queremos páginas enormes com acesso rápido

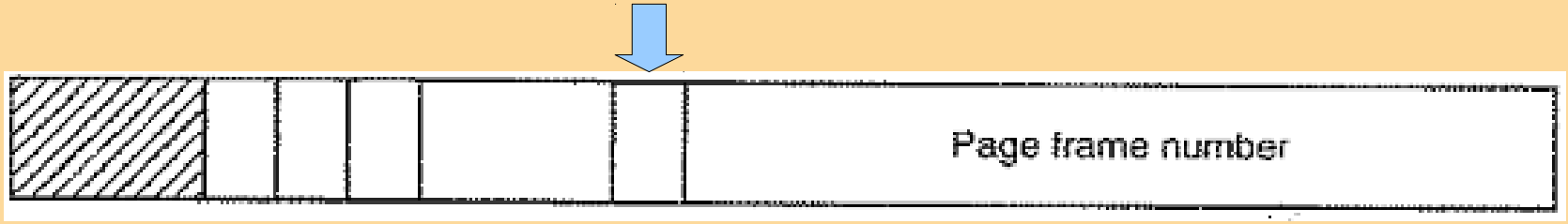
Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Depende muito do hardware
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Page frame number:
      - Identifica a página real
      - Campo mais importante



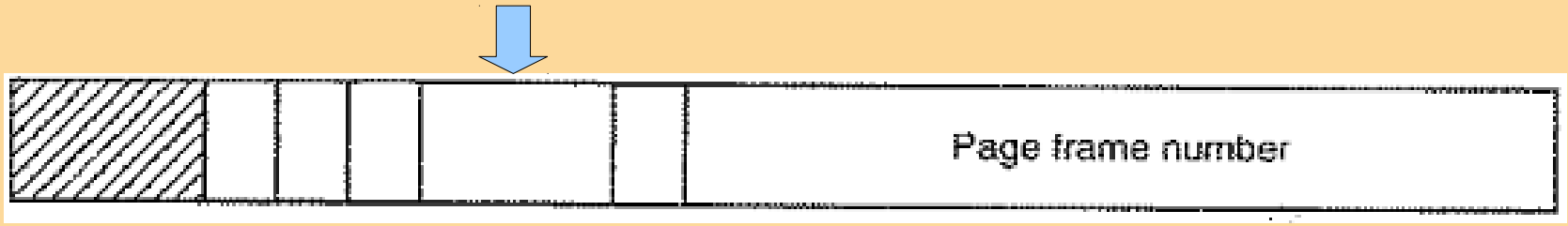
Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Bit de Residência (Presente/ausente):
      - Se valor igual 1, então entrada válida para uso;
      - Se valor igual 0, então entrada inválida, pois página virtual correspondente não está na memória (acessá-la causará page fault) – veremos mais adiante



Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Bits de Proteção:
      - Indicam tipos de acessos permitidos:
        1 bit → 0 – leitura/escrita
        1 – leitura
        3 bits → 0 – Leitura
        1 – Escrita
        2 – Execução

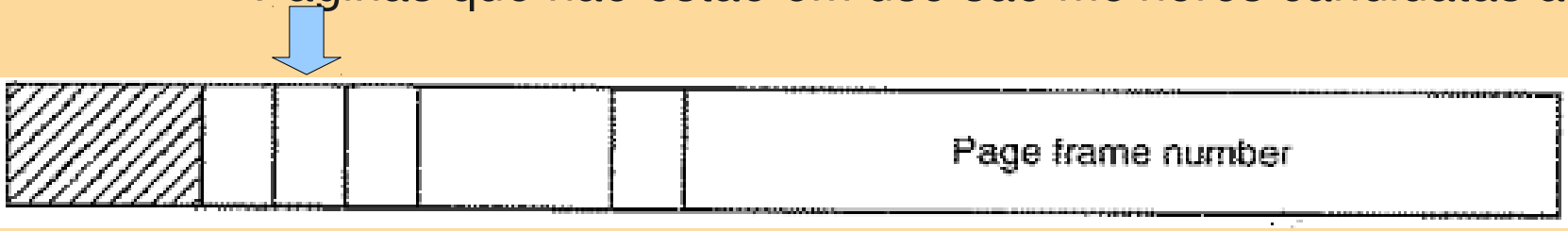


Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Bit de Modificação:
      - Controla o uso da página;
      - Se página foi escrita, valor igual a 1
        - Página deve copiada para o disco, caso seja removida da memória
      - Se valor igual a 0, página não foi modificada;
        - Página é abandonada, já que a cópia em disco é válida

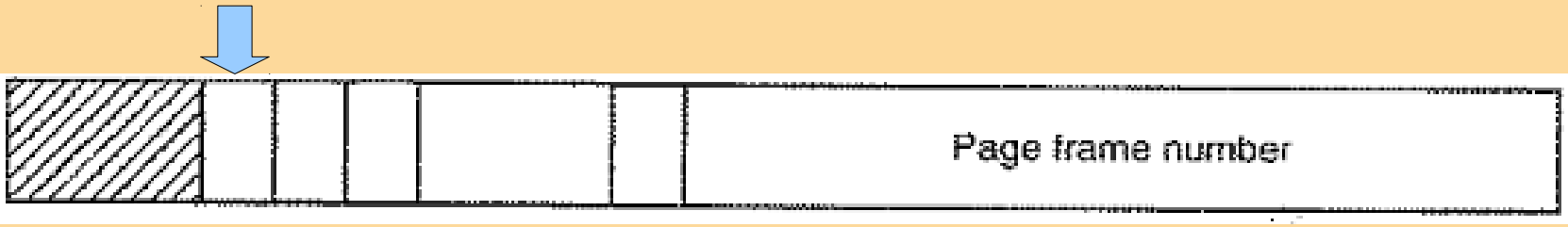
Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Bit de Referência:
      - Quando a página é referenciada (para leitura ou escrita), o hardware faz o bit ser 1
      - Em um dado intervalo de tempo, uma interrupção do clock faz o bit ser 0 (para todas as entradas)
        - Apenas páginas referenciadas dentro do intervalo de clock são marcadas
      - Auxilia o SO na escolha da página que deve deixar a RAM, em caso de page fault
        - Páginas que não estão em uso são melhores candidatas a sair



Implementação da Tabela de Páginas

- Entrada na Tabela de Páginas:
  - Em geral, 32 bits, divididos da seguinte maneira:
    - Bit de Cache:
      - Permite desabilitar o caching para a página
      - Necessário para páginas que mapeiam a registradores de dispositivos, em vez da memória
        - O hardware deve acompanhar diretamente o dispositivo, e não usar uma cópia em cache



Implementação da Tabela de Páginas

- A tabela de páginas pode ser armazenada de três diferentes maneiras:
  - Array de Registradores, se a memória for pequena;
    - Mantidos no hardware
  - Na própria memória RAM
    - A MMU gerencia utilizando um ou dois registradores
  - Em uma memória cache na MMU chamada Memória Associativa
    - Usada para melhorar o desempenho da tabela na RAM

Implementação da Tabela de Páginas

- Tabela de página mantida na memória principal
  - Possui 2 registradores associados:
    - Registrador de base da tabela de página (PTBR)
      - Page table base register
      - Aponta para o início da tabela de página, indicando o endereço físico de memória onde a tabela está alocada;
    - Registrador de tamanho da tabela de página (PTLR)
      - Existente apenas em alguns sistemas
      - Page-table length register
      - Indica tamanho da tabela de página (número de entradas da tabela → número de páginas);

Tabela de Páginas na RAM

- Nesse esquema, cada acesso de dado/instrução exige dois acessos à memória:
  - Um para a tabela de página e um para o dado/instrução
  - Cada acesso à memória, feito no programa, se transforma em 2
  - Contudo, muitos programas tendem a fazer um grande número de referências a um pequeno número de páginas
    - Apenas uma fração pequena das entradas na tabela são lidas com grande freqüência
    - O que fazer?

Tabela de Páginas na RAM

- Solução:
  - O problema dos dois acessos à memória pode ser solucionado pelo uso de um cache de hardware especial para pesquisa rápida
    - Chamado memória associativa ou translation lookaside buffers (TLBs)
    - Hardware especial para mapear endereços virtuais para endereços reais sem ter que passar pela tabela de páginas na memória principal;

Estrutura de um TLB

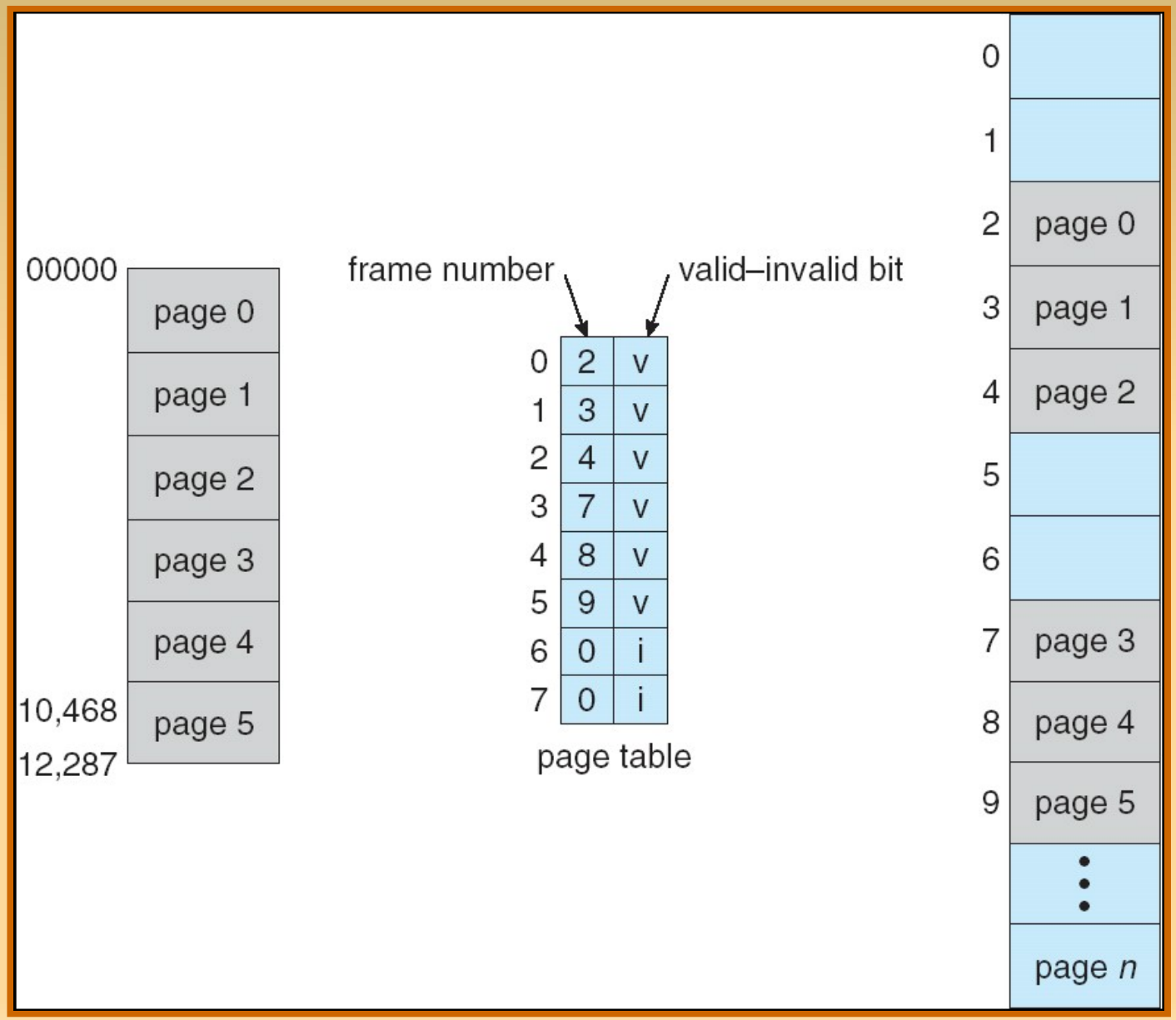
Valido = 1 → a página está em uso
Válido = 0 → não está em uso

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Proteção de memória

- Implementada associando-se bits de proteção e de válido/inválido a cada moldura
- Bit de válido-inválido anexado a cada entrada na tabela de página:
  - “válido” indica que a entrada na TLB possui dados válidos
  - “inválido” indica que a entrada ainda não foi usada
    - Necessário porque a TLB tem tamanho fixo, então há que se saber o que está em uso e o que não está

Bit de Válido/Inválido



O programa usa menos páginas que as entradas disponíveis na TLB → marca-se algumas inválidas

Memória Associativa (TLB)

Número da página virtual

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Com exceção desse valor, todos os demais também estão na tabela de páginas

Memória Associativa (TLB)

Feito 1 quando a página é modificada

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Memória Associativa (TLB)

Código de proteção: permissões para leitura/escrita/execução
RW → ex: variáveis, pilha de execução etc
RX → ex: código do programa a ser executado

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

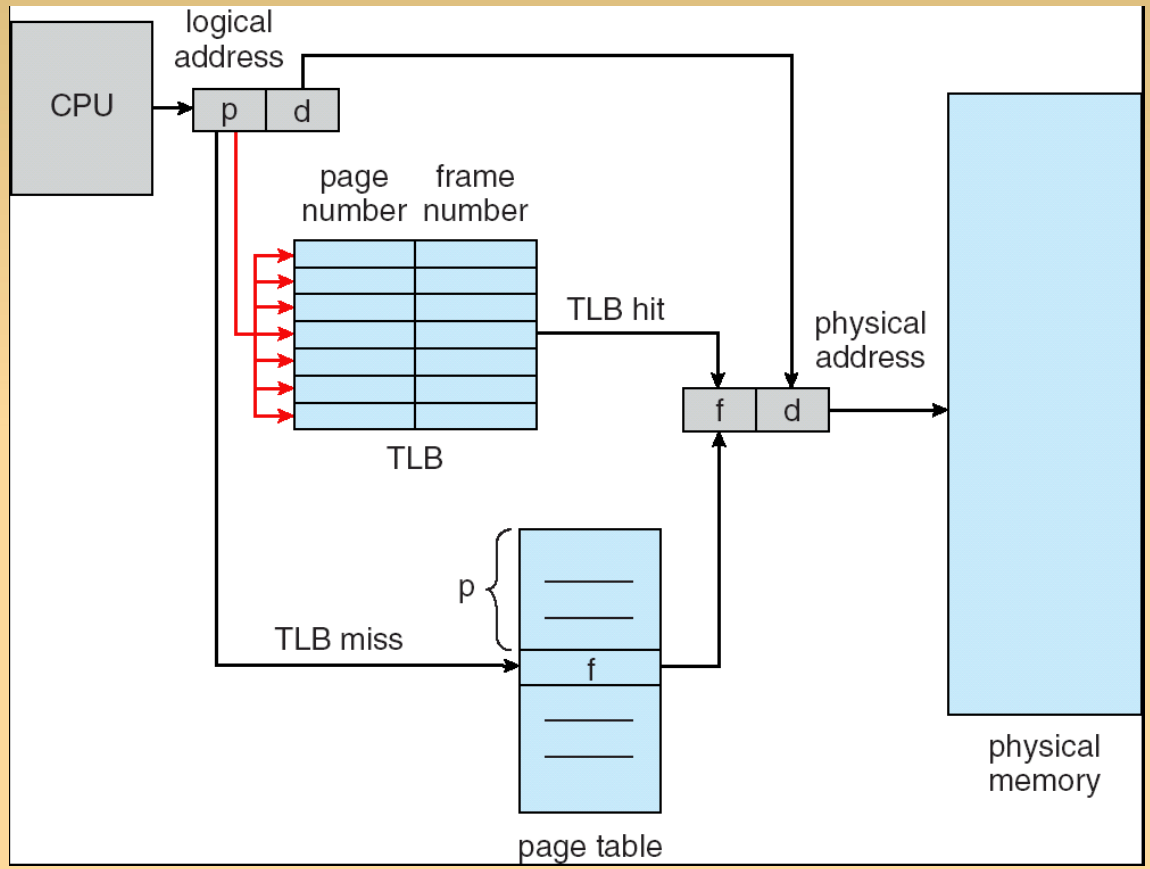
Memória Associativa (TLB)

Moldura física na qual a página está localizada

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

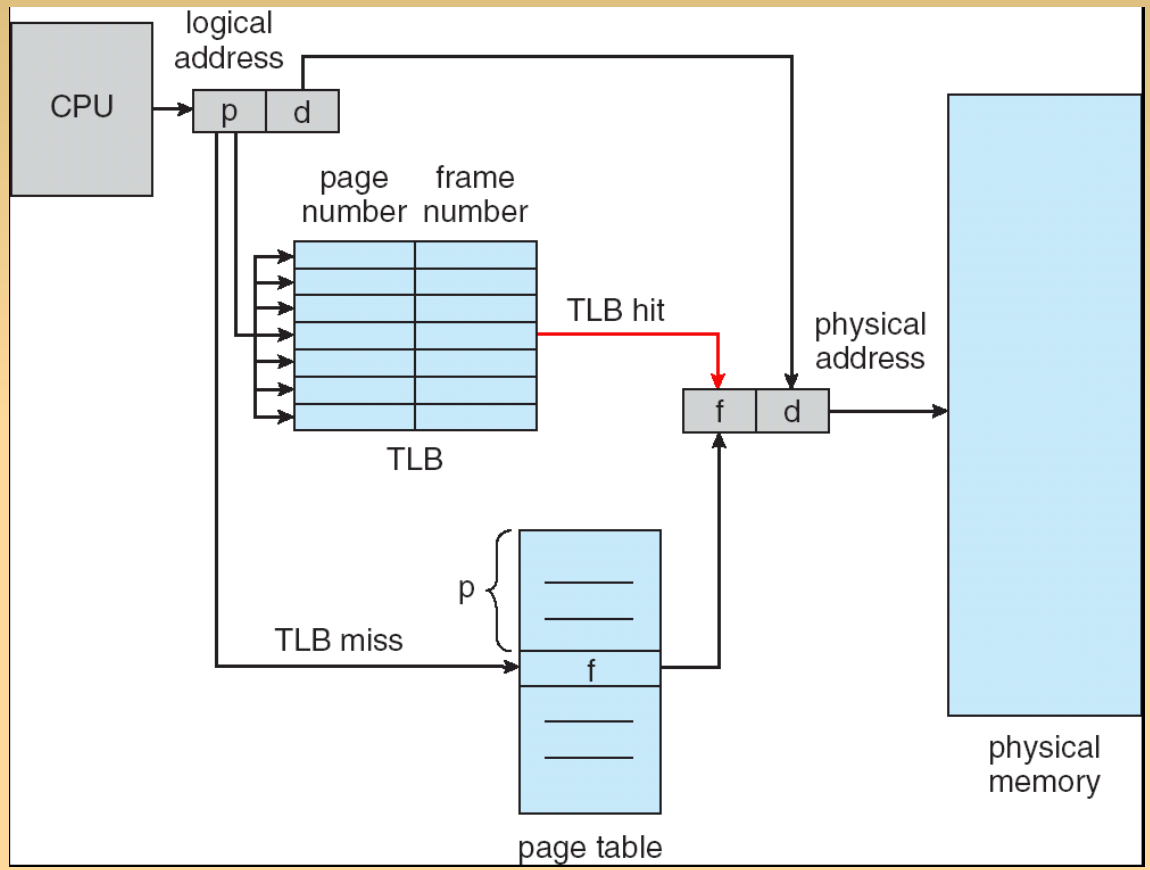
Memória Associativa (TLB)

- Funcionamento:
  - Quando um endereço virtual chega à MMU, o hardware verifica se sua página virtual está na TLB
    - Compara a todas as entradas simultaneamente (operação feita no hardware)



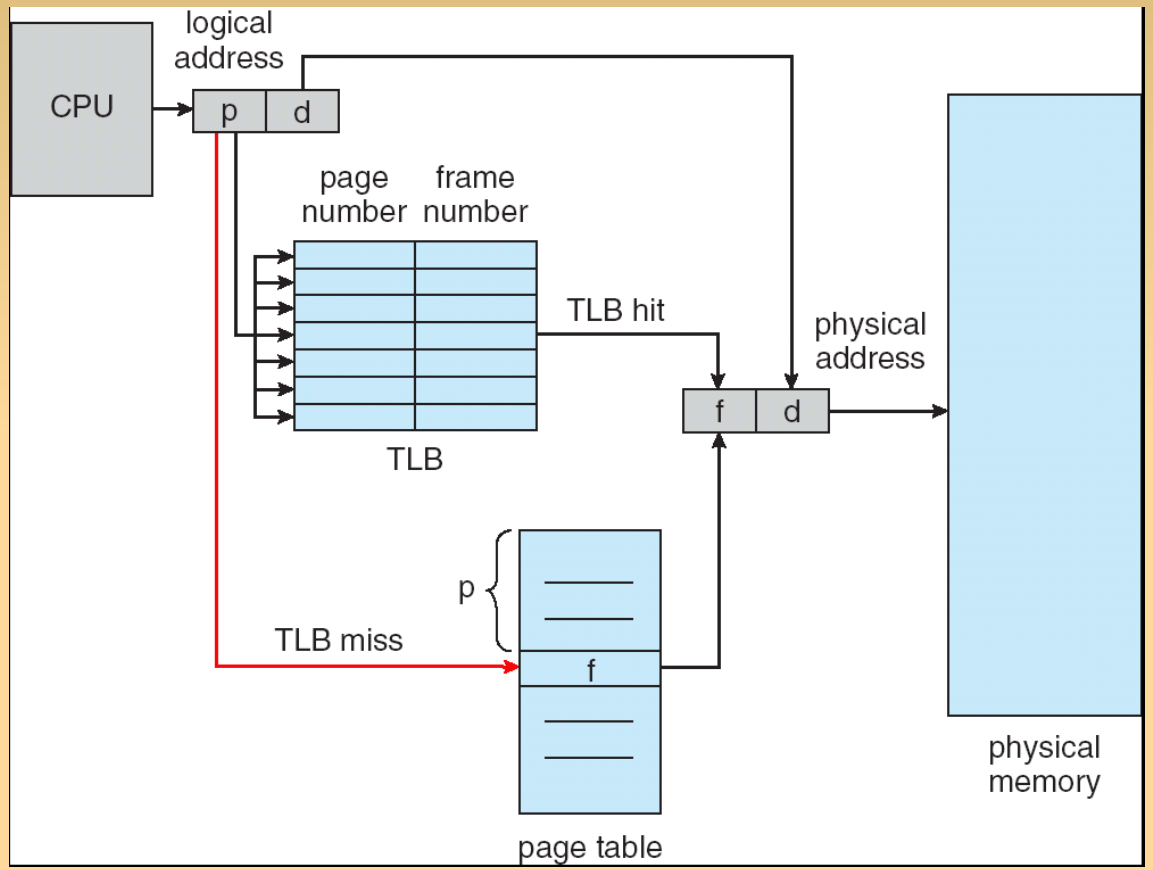
Memória Associativa (TLB)

- Funcionamento:
  - Se estiver na TLB (hit), e os bits de proteção não forem violados, a moldura é obtida da TLB, sem passar pela tabela de páginas
  - Se violar algum dos bits de proteção é gerada uma falha de proteção (protection fault)
    - Desvio ao S.O. via trap



Memória Associativa (TLB)

- Funcionamento:
  - Se a página virtual não estiver na TLB
    - A MMU busca-a na tabela de páginas
    - Remove uma das entradas da TLB, devolvendo-a à tabela na memória
    - Coloca nessa entrada a página que acabou de buscar
    - Se essa página for usada novamente, estará na TLB



Memória Associativa (TLB)

- Pode ser implementada em:
  - Hardware
    - Maior rapidez
    - Ocupa espaço físico que poderia ser disponibilizado para outras funções (cache etc)
  - Software
    - Usa o software SO (mais lento); gera traps
    - As páginas contendo a tabela de páginas podem não estar na TLB, quando do processamento de uma page fault
      - Causará novas page faults durante o processo

Memória Associativa (TLB)

- Dois tipos de falhas em encontrar páginas:
  - Soft miss
    - Quando a página referenciada não está na TLB, mas está na memória física
    - Basta atualizar a TLB
  - Hard miss
    - A página em si não está na memória física (e nem na TLB, naturalmente)
    - Deve-se fazer um acesso ao disco para trazê-la à memória (e então à TLB)
    - Muito lento