INE5412 Sistemas Operacionais I

L. F. Friedrich

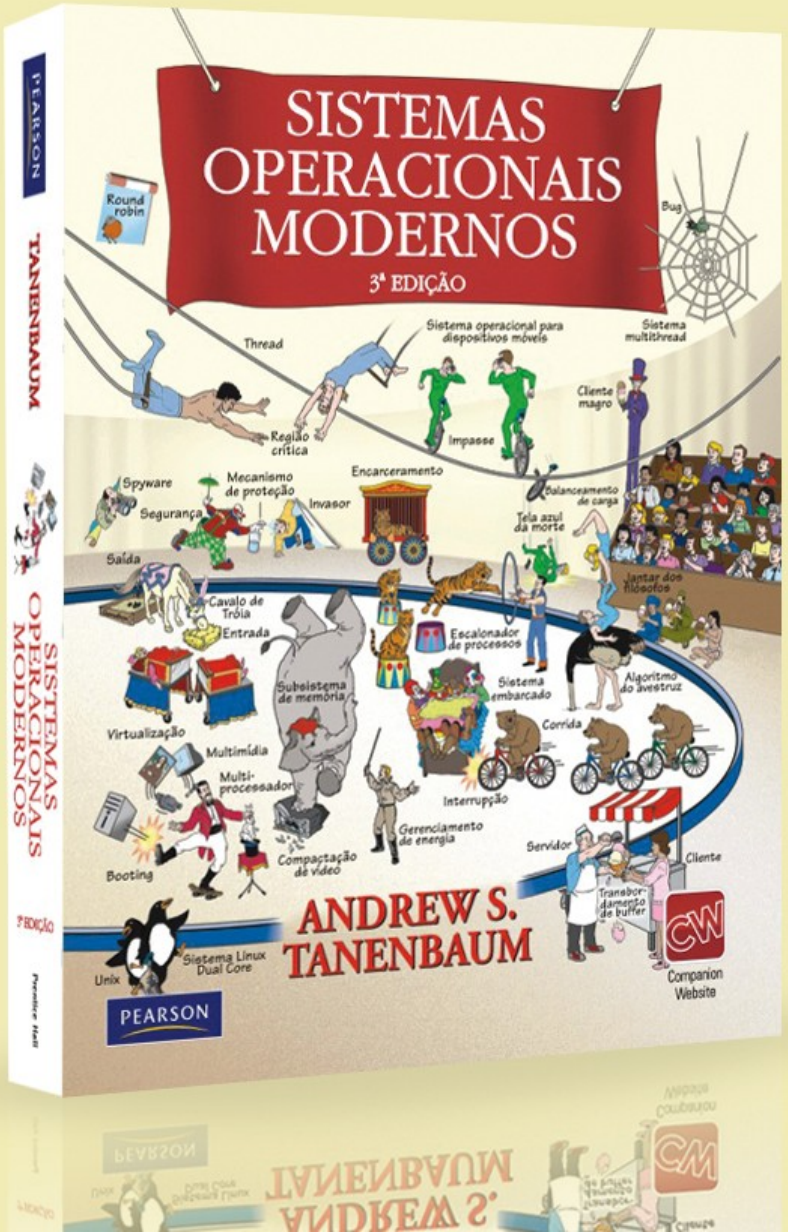
Capítulo 3

Memoria Virtual : Paginação

Sistemas operacionais modernos
Terceira edição

ANDREW S. TANENBAUM

Capítulo 3
Gerenciamento de memória



Memória virtual - paginação

A princípio o mecanismo baseado em registradores-base e registradores-limite nos permitem criar a abstração de **espaços de endereçamento** com os conceitos de **endereço virtual** (lógico) e **endereço físico**.

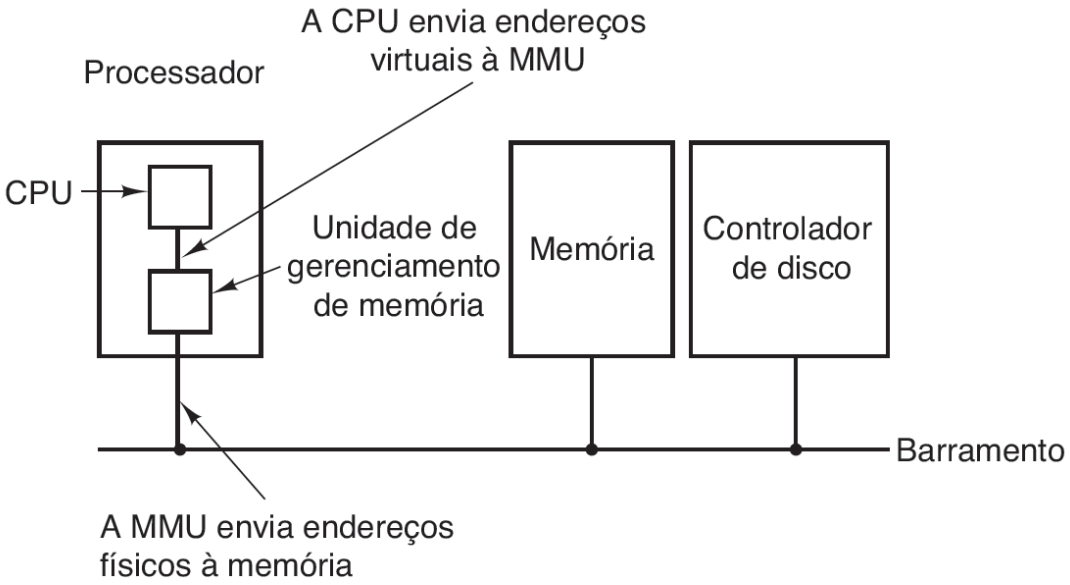


Figura 3.8 A posição e a função da MMU. Aqui a MMU é mostrada como parte do chip da CPU (processador) porque isso é comum atualmente. Contudo, em termos lógicos, poderia ser um chip separado, como ocorria no passado.

Memória virtual - paginação

- Do ponto de vista de alocação de memória, a abordagem de **alocação contígua** associada a um mecanismo de **swapping** permite que o Sistema Operacional possa gerenciar de forma satisfatória a demanda de memória por parte dos processos (programas) a serem executados.
- Entretanto, é necessário que o processo (programa) esteja totalmente residente na memória para que o mesmo possa ser executado.
- Esta restrição traz outros problemas que devem ser resolvidos, entre eles:
  - Fragmentação externa
  - Impossibilidade de executar programas maiores que a memória disponível

  (Bloatware -http://www.webopedia.com/TERM/B/bloatware.html)

Paginação

- O problema da fragmentação tem como solução a utilização do conceito de mapeamento da memória.
- Com o auxílio de um dispositivo de MMU, o espaço de endereçamento do processo é mapeado na memória física.
- Endereços lógicos são mapeados para Endereços físicos
  - O espaço de endereçamento do processo é dividido em unidades denominadas **páginas (Paginação)**

Paginação

- Espaço de endereço de um processo pode ser não contíguo; ao processo é alocado memória física sempre que disponível.
- Divide *memória física* em blocos de tamanho fixo chamados de ***frames-molduras de páginas*** (tamanho é potência de 2, entre 512 bytes- 8192 bytes).
- Divide *memória lógica* em blocos de mesmo tamanho chamados de ***páginas***.
- Mantém informação sobre todos ***frames*** livres
- Para executar um programa de tamanho ***n páginas***, necessário encontrar ***n frames*** livres e carregar o programa.
- Sistema Operacional - Prepara uma tabela de páginas para traduzir endereços lógicos em físicos.

Paginação simples

Exemplo de atribuição de páginas de processo a frames.

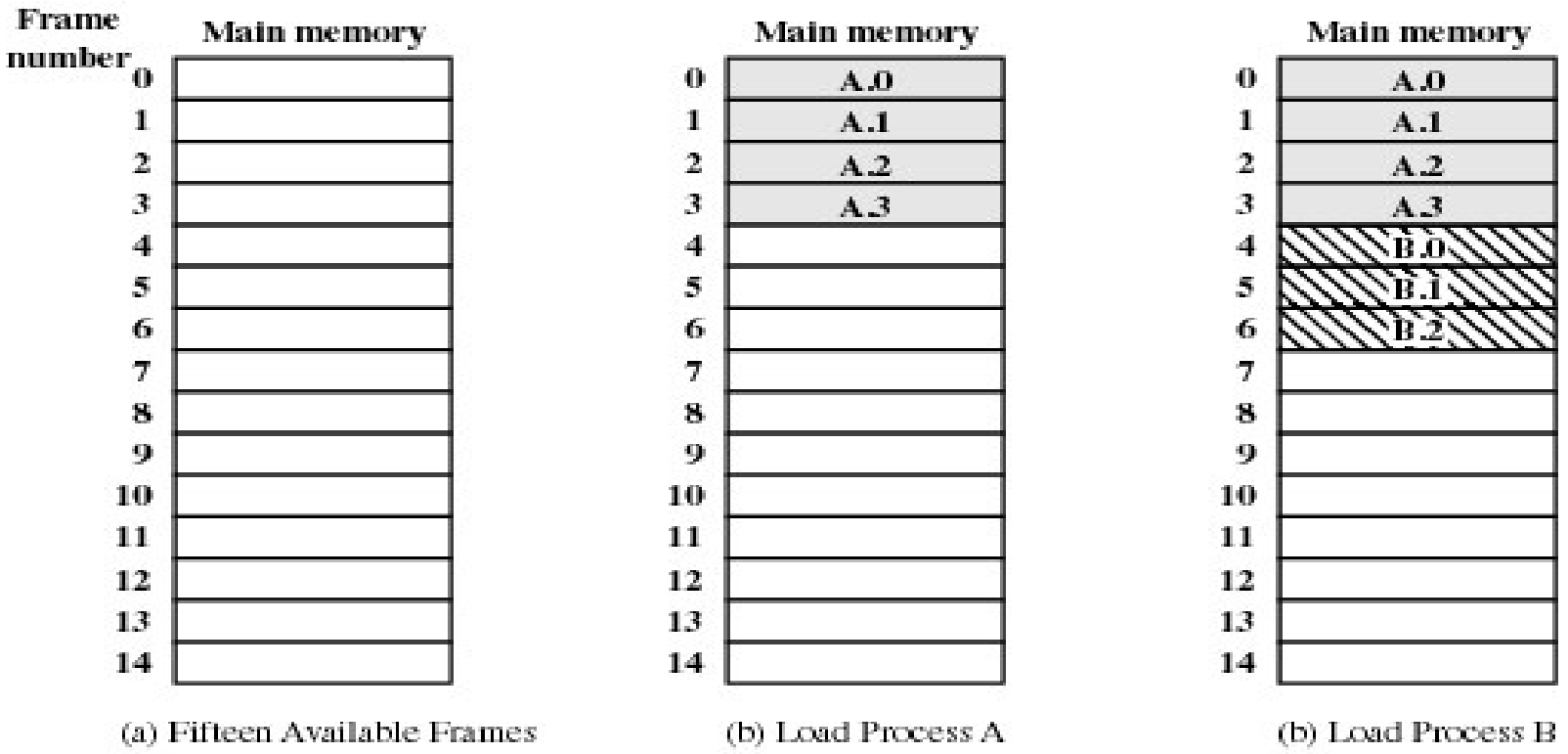


Figure 7.9 Assignment of Process Pages to Free Frames

Paginação simples

Exemplo de atribuição de páginas de processo a frames.

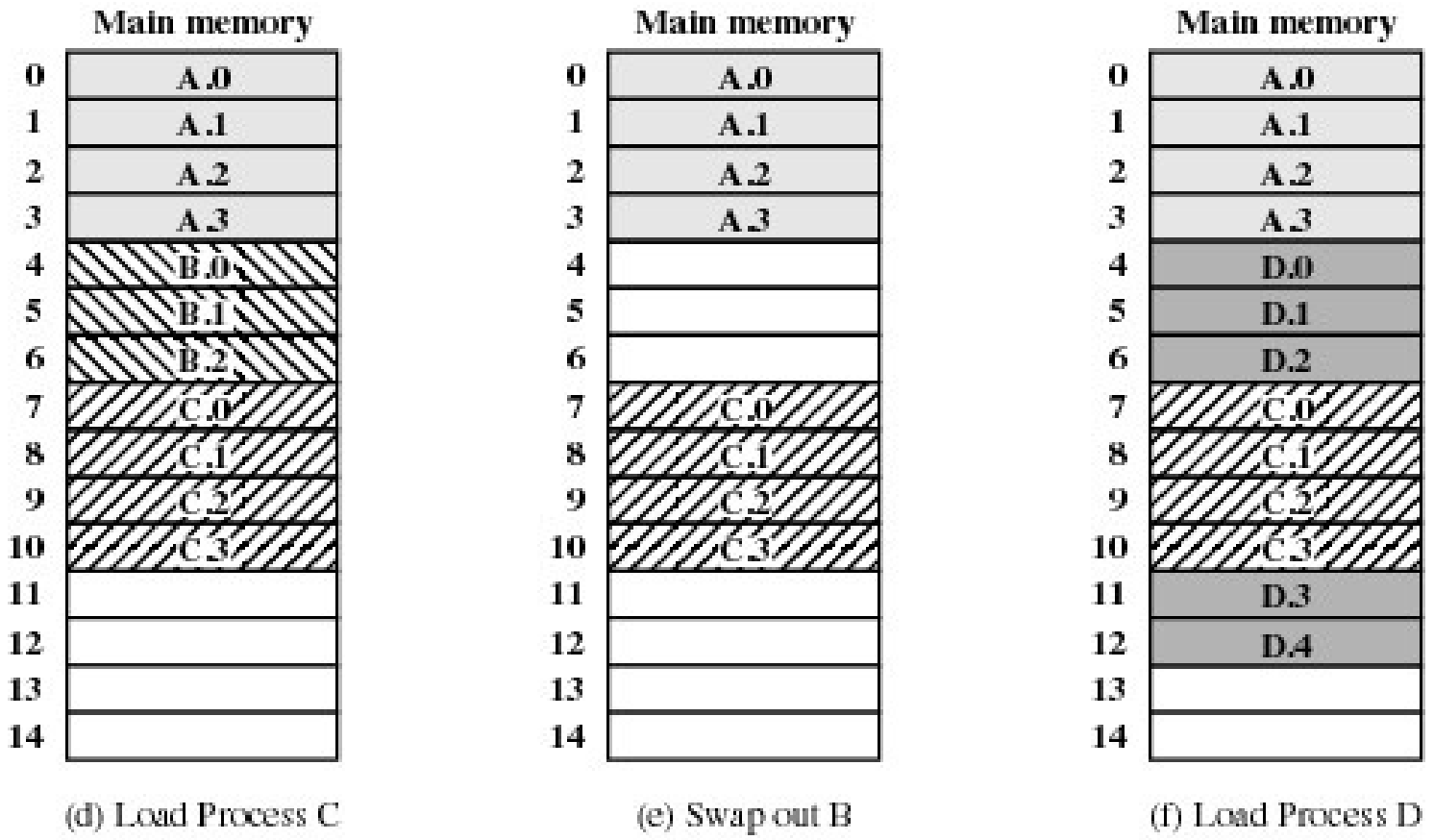


Figure 7.9 Assignment of Process Pages to Free Frames

Paginação simples

Exemplo de tabela de página para processos:

- contém a localização do **frame** para cada **página** do processo
- objetivo é mapear páginas lógicas em molduras de páginas físicas
- é uma função que usa o numero da página lógica como argumento e tem o número da moldura de página física correspondente como resultado.

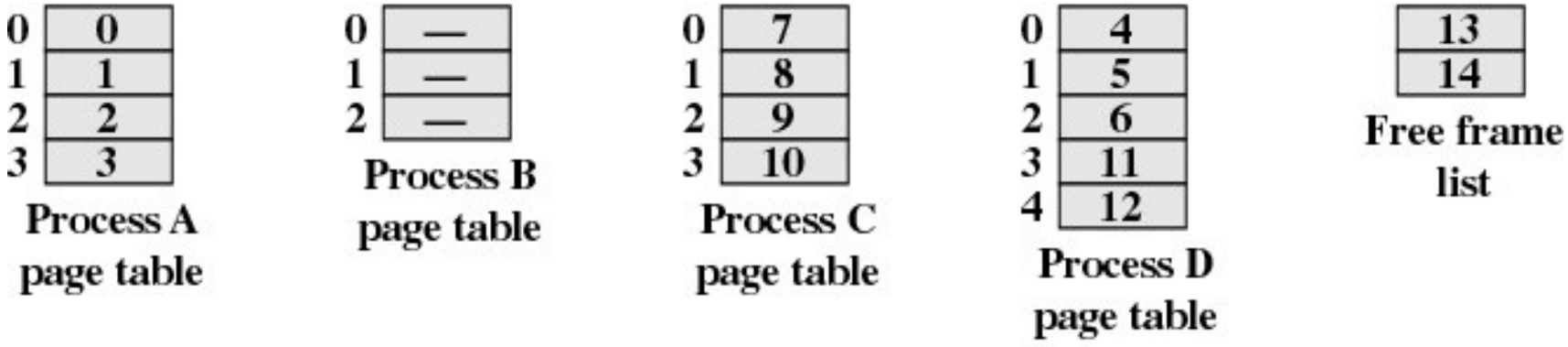


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Paginação simples

Endereços lógicos (virtual) são formados por:

- o numero da **página**
- o deslocamento (**offset**) dentro da **página**

Exemplo: endereço de 16 bits e página de 4KB -

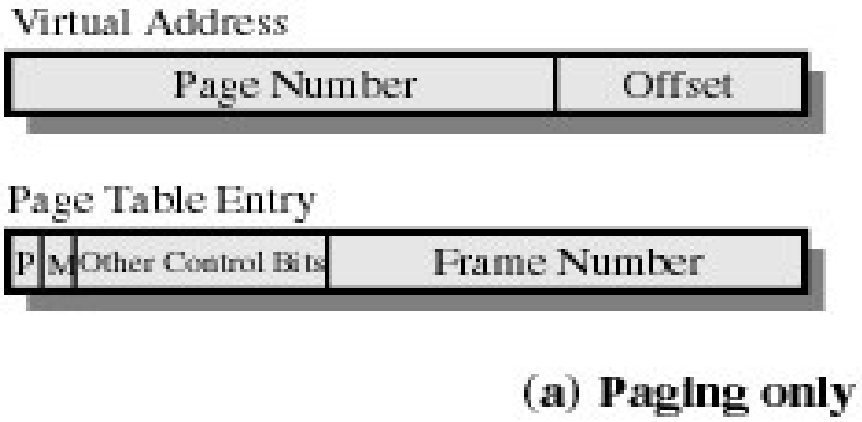


Figure 8.2 Typical Memory Management Formats

Paginação simples - tradução

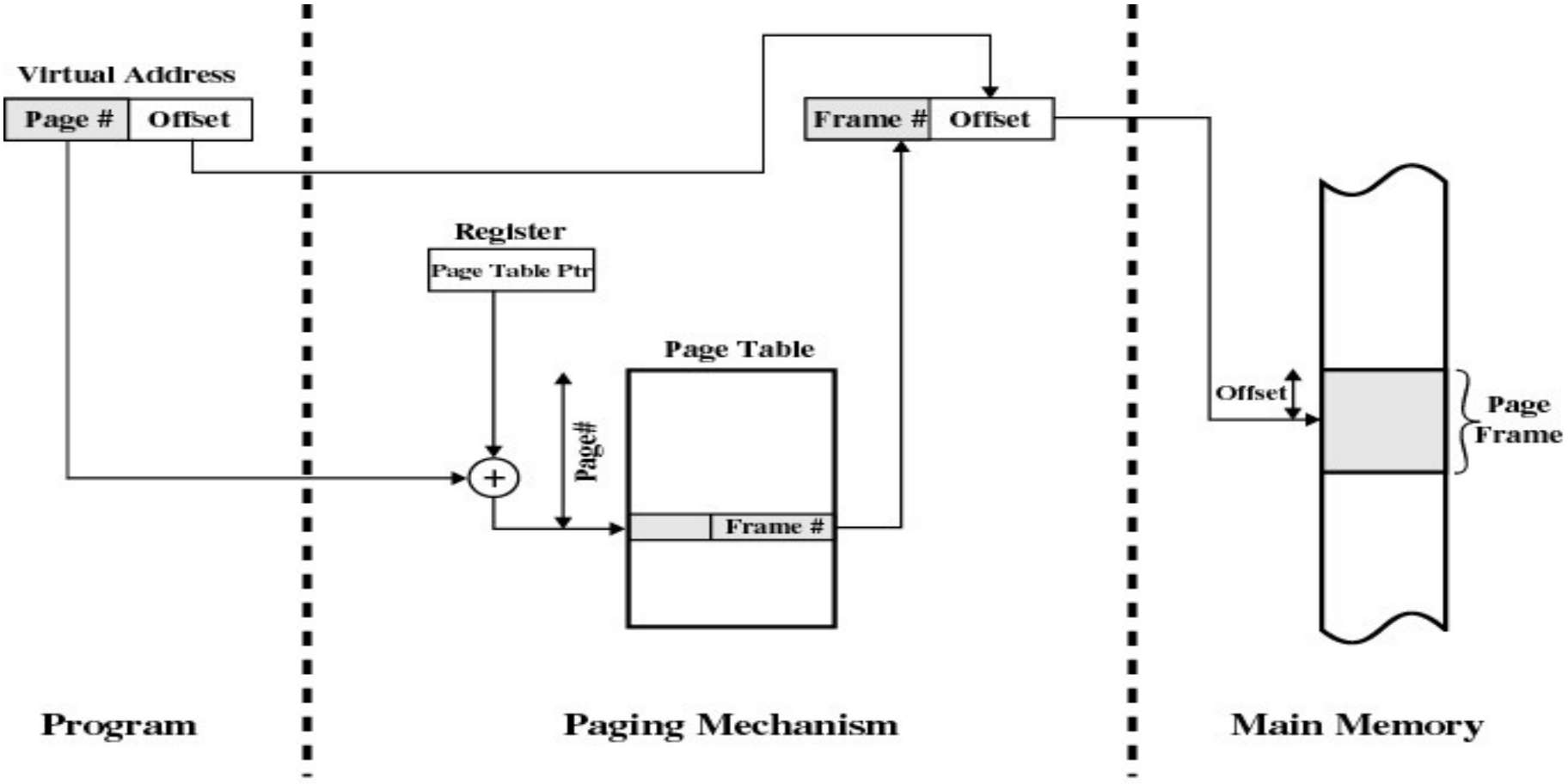
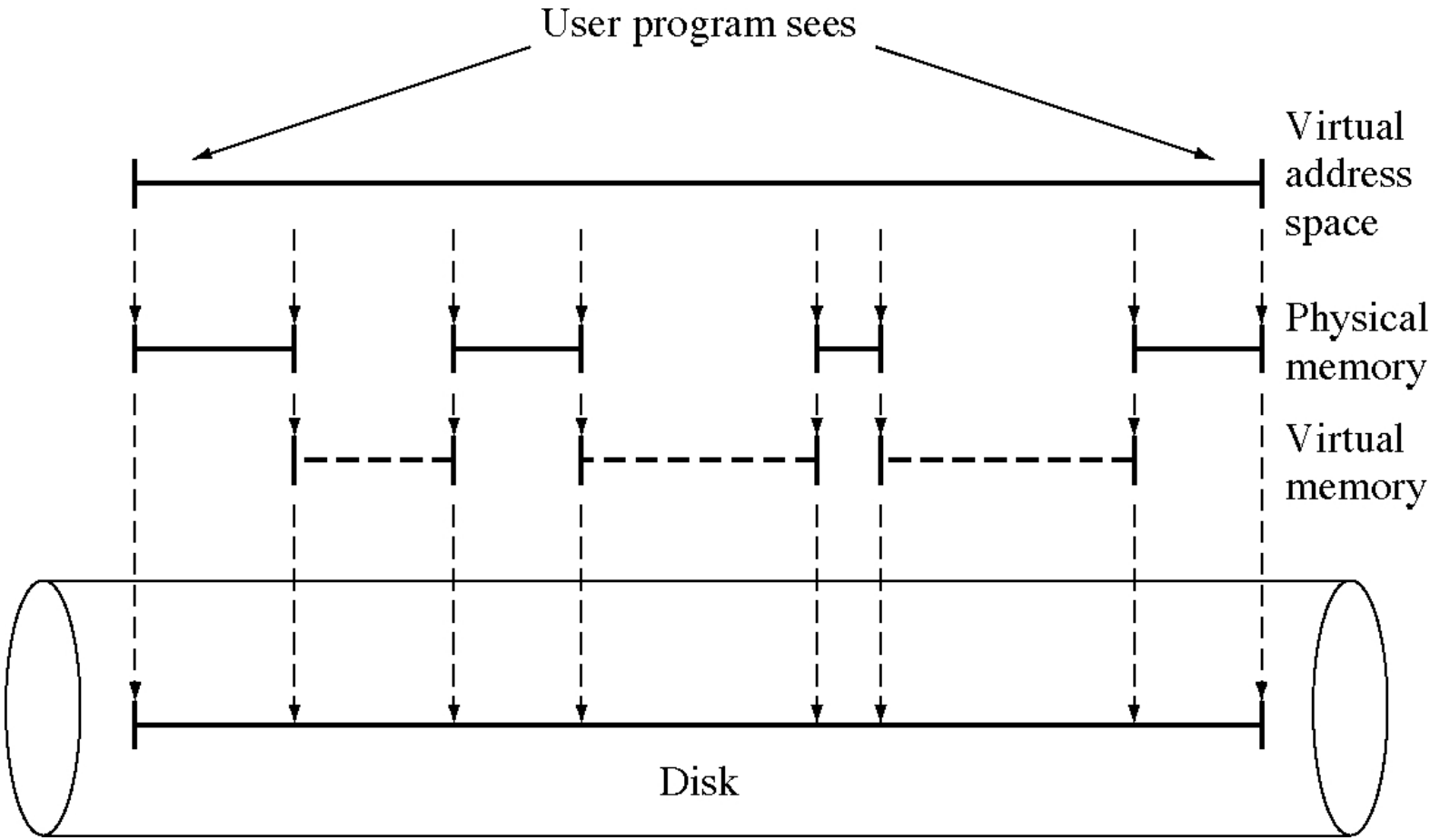


Figure 8.3 Address Translation in a Paging System

Memória Virtual

- O problema de programas maiores que a memória está presente desde o início da computação.
- Uma solução adotada – divisão do programa em módulos – conhecida como ***overlay*** (sobreposição), onde os módulos eram carregados de forma sobreposta(0,1,..). Divisão do programa feito pelo programador.
- Fotheringham,1961, método conhecido como **memória virtual.**
- Idéia: cada programa (processo) tem seu próprio espaço de endereçamento, que é dividido em blocos chamados **páginas**.

Memória Virtual



Memória Virtual

- **Memória Virtual permite:**
  - Multiplexação da memória no tempo
  - Apenas parte do programa precisa estar na memória para execução.
  - EEL pode ser maior que EEF.
  - EE ser compartilhado por vários processos.
  - criação de processos mais eficiente.

- Memória virtual pode ser implementada via:
  - Paginação por Demanda
  - Segmentação por Demanda

Paginação por Demanda

- Paginação por Demanda busca uma página do disco para a memória física quando aquela página é demandada.
  - Similar ao swapping, mas o sistema não traz todo o processo para a memória, apenas as paginas requeridas.
  - Isto permite que mais processos sejam hospedados no sistema.

Memória Virtual

Ex: endereços virtuais de 16bits, sendo que a capacidade física é 32KB. O programa pode ter até 64KB mas não é possível carregar totalmente. No exemplo as páginas tem 4K (512 a 64KB) –

- 16 páginas e 8 frames.
- MOV REG,0 – (8192)
- MOV REG, 8192 – (24576)

As páginas com X não estão mapeadas:

- MOV REG,32780
- MMU identifica não mapeamento
- isto gera uma PAGE FAULT
- SO escolhe um frame
- SO carrega a página virtual faltante
- SO atualiza mapeamento na TP
- SO reinicializa instrução causadora

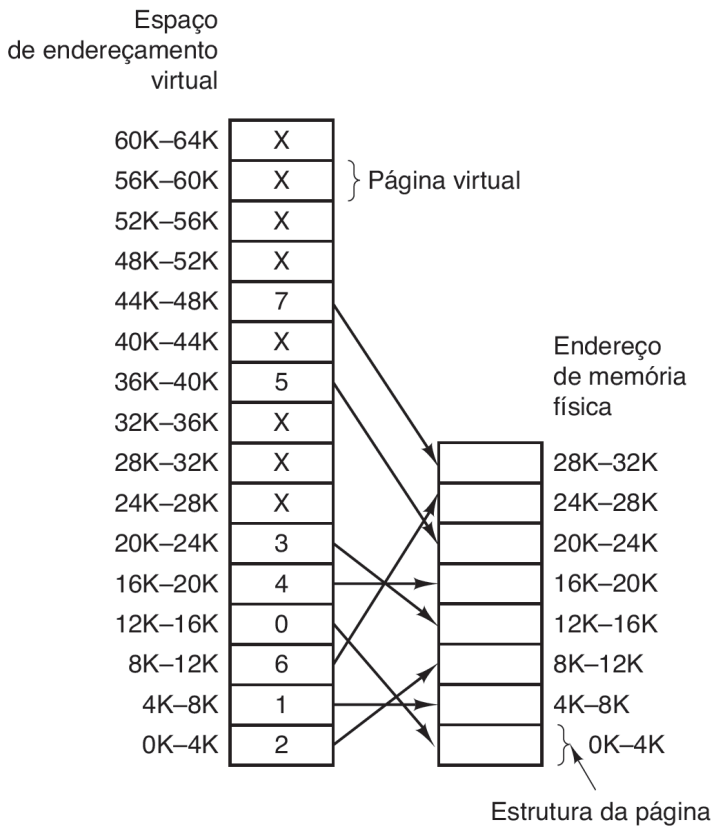


Figura 3.9 A relação entre endereços virtuais e endereços de memória física é dada pela tabela de páginas. Cada página começa com um múltiplo de 4096 e termina 4095 endereços acima; assim, 4K–8K na verdade significa 4096–8191 e 8K–12K significa 8192–12287.

Memória Virtual

Exemplo de funcionamento da MMU, no caso anterior:
EV: 8196
0010 0000 0000 0100
Pag. Deslocamento

Bit presente/ausente indica se a mesma esta na memória (1) ou não (0)
(1) numero do frame, 3bits, é concatenado com 12 bits do deslocamento, formando o endereço físico de 15 bits.

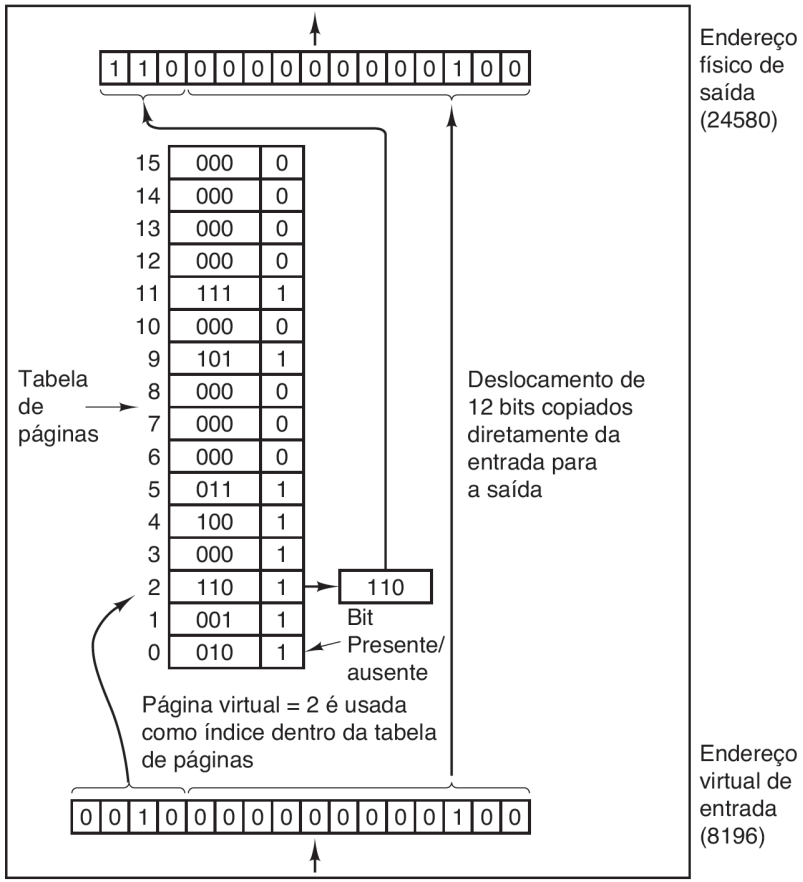


Figura 3.10 Operação interna da MMU com 16 páginas de 4 KB.

Tabela de páginas

- Uma tabela de páginas armazena mapeamento de memória.
  - A tabela diz quais páginas estão na memória física.
  - A tabela é armazenada na MMU da UCP (ideal).
- O conteúdo da tabela de páginas depende do mecanismo de paginação.
  - Numero do frame, bits de proteção, modificação, referência,

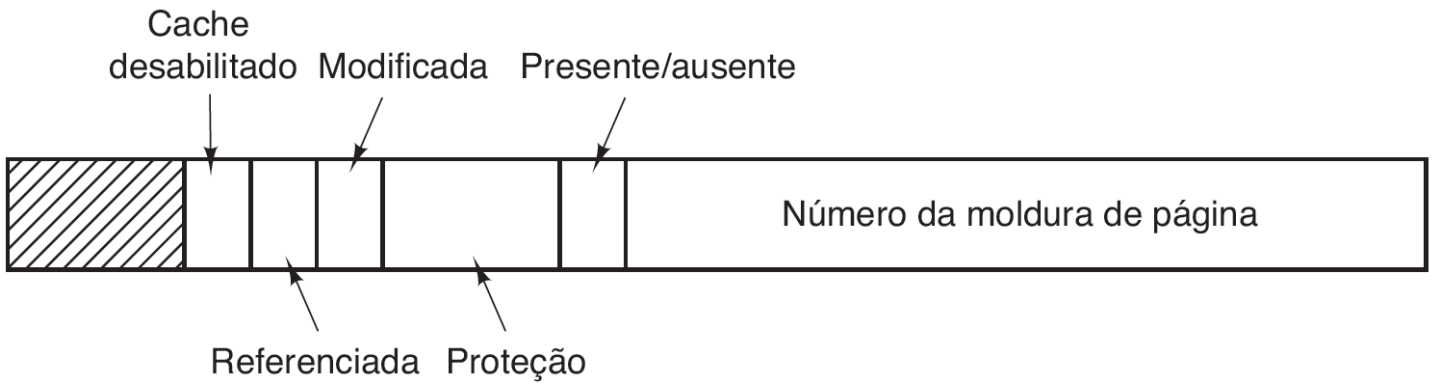


Figura 3.11 Entrada típica de uma tabela de páginas.

MV: algoritmo (1 of 2)

```
const int LogicalPages  = 1024;
const int BytePerPage  = 4096;
const int OffsetShift   = 12;
const int OffsetMask    = 0xFFF;
const int PhysicalPages =  512;
enum AccessType { invalid = 0, read = 1, write =
2, execute = 3 };
struct PageTableEntry {
  int pageBase   : 9;
  int present    : 1;
  AccessType protection : 2;
  int fill       : 4; // fill to 16 bits
};
PageTableEntry UserPageTable[LogicalPages];
```

MV: algoritmo (2 of 2)

```
int MemoryAccess( int logicalAddress,
    AccessType how, int dataToWrite = 0 ) {
  int page = logicalAddress >> OffsetShift;
  int offset = logicalAddress & OffsetMask;
  PageTableEntry pte = UserPageTable[page];
  if( how != pte.protection )
    if( !(how = read && ptr.protection = write) ) {
      CauseInterrupt( ProtectionViolation );
      return 0;}
  if( pte.present == 0 ) {
    GenerateInterrupt( PageFault, page );
    return 0; }
  int physicalAddress
    = (pte.pageBase << OffsetShift) + offset;
  switch( how ) {
    case read: case execute:
      return PhysicalMemoryFetch(physicalAddress);
    case write:
      PhysicalMemoryStore(
        physicalAddress, dataToWrite );
      return 0;
  }
```

Acelerando a Paginação

- Apresentamos os princípios básicos da Paginação e memória virtual. Vamos ver implementações possíveis. Dois problemas são importantes:
  - O mapeamento do endereço virtual para endereço físico deve ser rápido.
    - Mapeamento feito a cada referência
  - Se o espaço de endereço virtual for grande , a tabela de páginas será grande.
    - Endereços virtuais hoje: 32bits, 64bits
    - Cada processo precisa sua tabela

Acelerando a Paginação

- Projeto mais simples :
  - Tabela de páginas é composta por conjunto de registradores (Fig. 3.10).
    - SO carrega os registradores com a tabela de páginas (mantida em memória)
    - Vantagem: direto, sem referências a memória
    - Desvantagem: caro, carga da tabela completa a cada chaveamento
  - Tabela de páginas na memória principal, 1 registrador para apontar tabela.
    - Mapa modificado carregando registrador
    - Uma ou mais referências a memória, lenta

Acelerando a Paginação

- Maioria dos esquemas usados para acelerar a paginação parte do posicionamento da tabela na memória.
  - Grande impacto no desempenho
    - Ex. instrução de 1byte (copia de reg1 para reg2)
    - O que acontece s/ paginação e c/ paginação?
  - Solução com base na seguinte observação:
    - Maioria dos programas tende a fazer um grande numero de referências a um mesmo pequeno conjunto de páginas virtuais
    - Solução: equipar os computadores com hardware para mapeamento sem passar pela TP.

Localidade

- Programas não acessam seus espaços de endereços de maneira uniforme
  - eles acessam a mesma posição várias vezes
  - referenciando um mesmo conjunto de páginas
- *Localidade espacial*: processos tendem a acessar a posição mais próxima daquela acessada
- *Localidade temporal*: processos tendem a acessar os mesmos dados várias vezes

TLB ou memória associativa

- Pequeno dispositivo para mapear endereços virtuais em físicos sem passar pela TP – TLB – buffer para tradução de endereços.

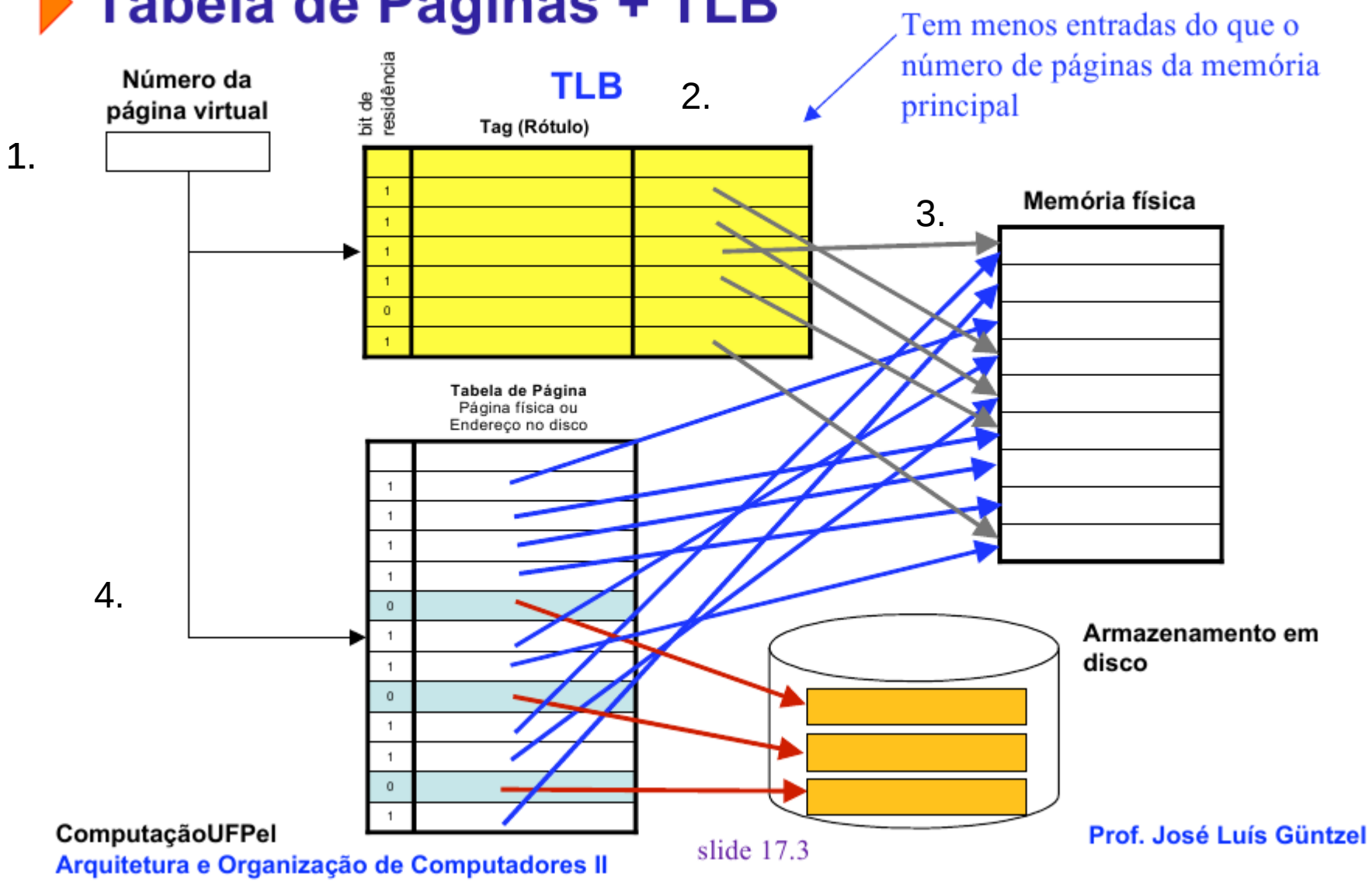
Localizado na MMU
Numero pequeno de entradas-8
Entrada contém info da página
Semelhante a TP
Ex: loop referenciando 19,20,21
Dados : 129, 130
Indices : 140
Pilha : 860 e 861

Quando EV é apresentado a MMU, verifica se página esta presente e o acesso é válido.

Válida	Página virtual	Modificada	Proteção	Moldura da página
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

Tabela 3.1 Uma TLB para acelerar a paginação.

Tabela de Páginas + TLB



TLB por software

- É suposto que sistema de MV (paginação) tem suporte completo do hardware da MMU, incluindo gerencia e tratamento de faltas. SO só é notificado qdo uma página não esta na memória.
- Hoje máquinas RISC (SPARC, MIPS, …) fazem a gerencia por software.
  - SO carrega entradas na TLB; MMU apenas gera int. de falta e repassa para o SO; SO deve realizar toda busca, troca e reinicialização; Isto é feito para muitas instruções.
  - Se TLB grande o suficiente (64), para reduzir a taxa de faltas, acaba tendo eficiência aceitável.
  - Ganho principal é ter uma MMU mais simples.
  - Estratégia para melhorar desempenho: tentativa de reduzir numero e custo das faltas.
    - Acessar TP por software, páginas que tem a TP podem não estar na TLB
  - Qdo TLB por software é usado, entender diferença entre *soft miss* e *hard miss*
    - Não esta na TLB mas na memória
    - Não está na TLB e não esta na memória

Estrutura da Tabela de Página

- TLBs aceleram a tradução de EV para EF em relação ao esquema original de TP. Além disso, é preciso resolver o problema de como lidar com EEV muito grandes. Dois modos são considerados:
  - Tabelas de Páginas multinível
  - Tabelas de Páginas Invertida

Tabela de Página Multinível

- Particiona espaço de endereço lógico em múltiplas tabelas de páginas.
- Um exemplo simples é tabela de página de dois níveis.
  - EV: 32bits – PT1(10), PT2(10) + desl. (12)
    - Página 4K, 2^{20} páginas virtuais
  - Evitar que todas as páginas sejam mantidas na memória o tempo todo.
  - Figura mostra funcionamento

Tabelas de páginas multinível

Suponha um processo: 12MB
- 4 código, base
- 4 dados
- 4 pilha, topo

Endereço 0x00403004 (4.206.596)

PT1 =?
PT2 =?
Deslocamento =?
Se bit presente=0?
Quantas tabelas são necessárias?
Expansão p/ 3 níveis

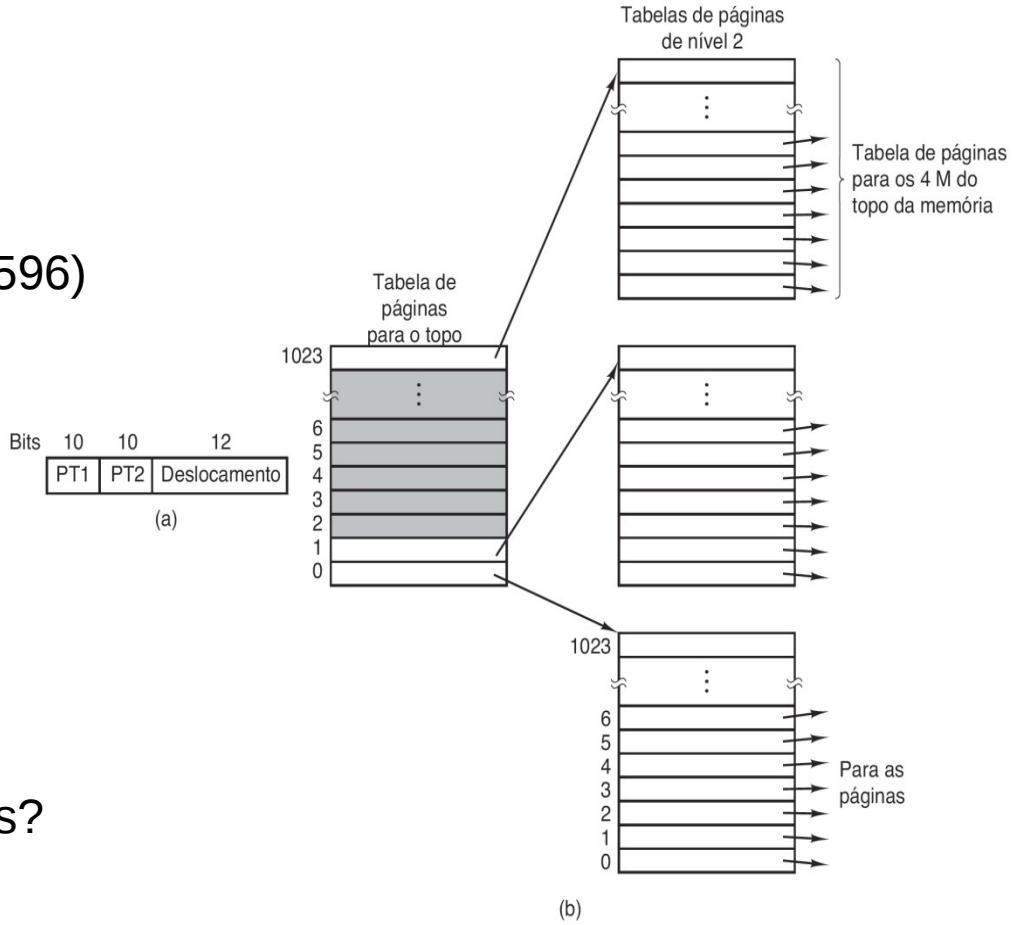


Figura 3.12 (a) Um endereço de 32 bits com dois campos de tabela de páginas. (b) Tabelas de páginas de dois níveis.

Tradução de endereços

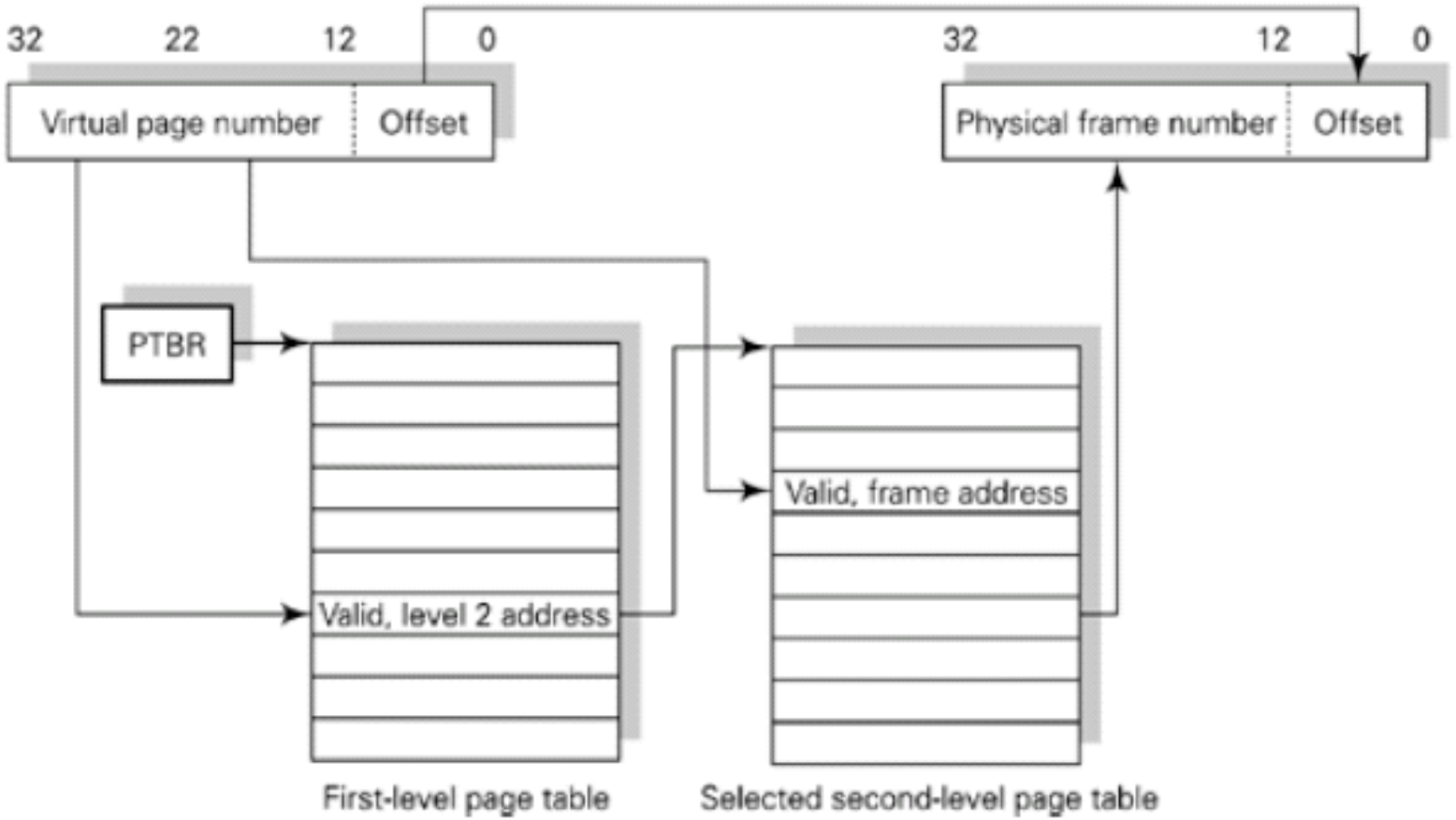


Tabela de Página Invertida

- Para EEV de 32 bits, a tabela de páginas multinível funciona razoavelmente bem. Com EEV de 64bits – 2^{64} .

  Páginas: 4k ---- 2^{52}
- Tabela de páginas invertida: Uma entrada para cada página real da memória.
- Entrada consiste de o endereço virtual da página armazenada na posição de memória, com informação sobre o processo que é dono da página.
- Diminuição da memória necessária para armazenar cada tabela de página,
- mas aumenta o tempo necessário para pesquisar a tabela quando uma referência a página ocorre – alternativa TLB – quando miss?
- Usar tabela de hash para limitar a pesquisa para uma — ou no máximo poucas — entradas da tabela de página.

Exemplo: EV – 64bits , páginas – 4MB → 2^{52}

Tabela de Página Invertida

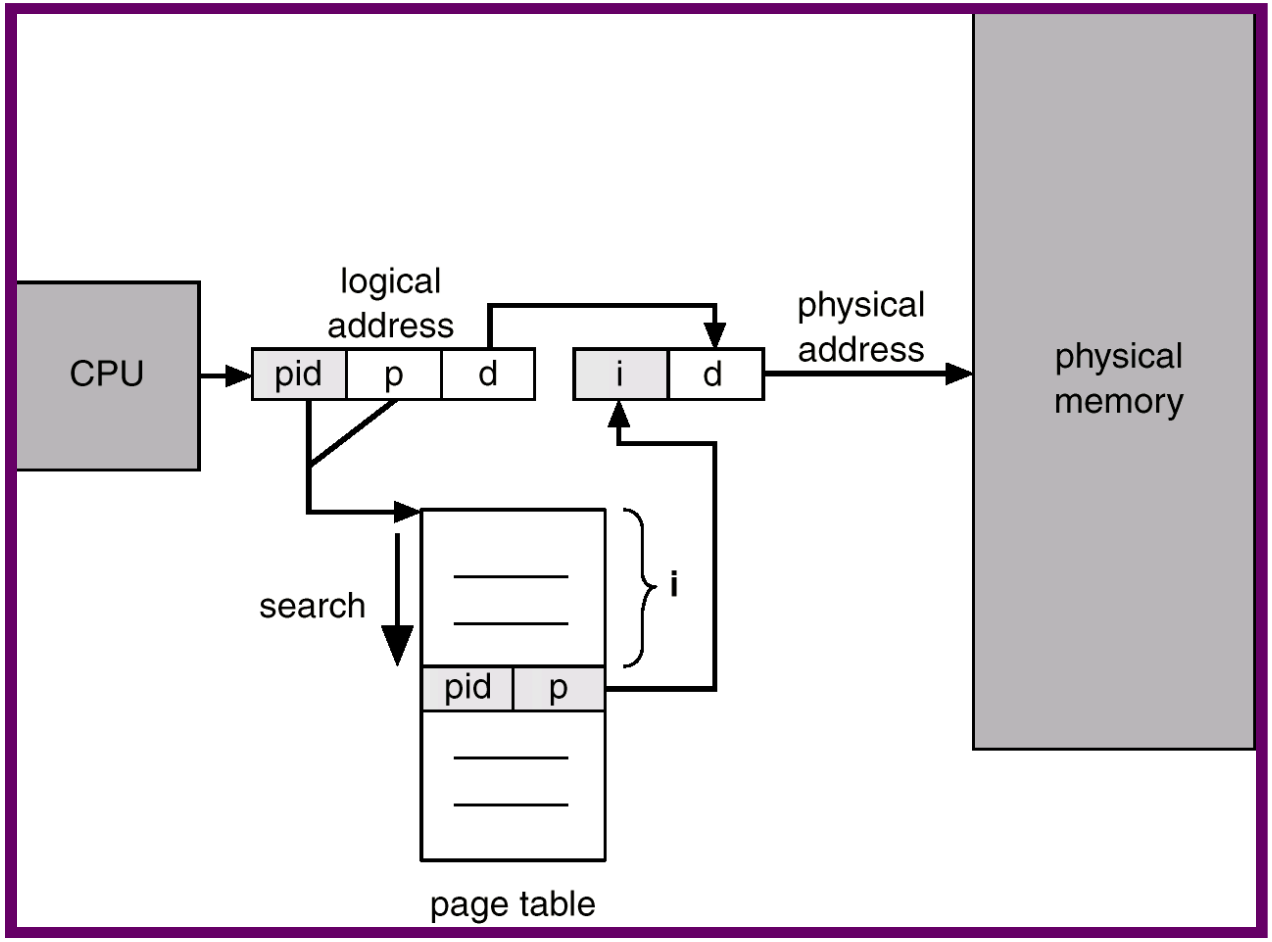
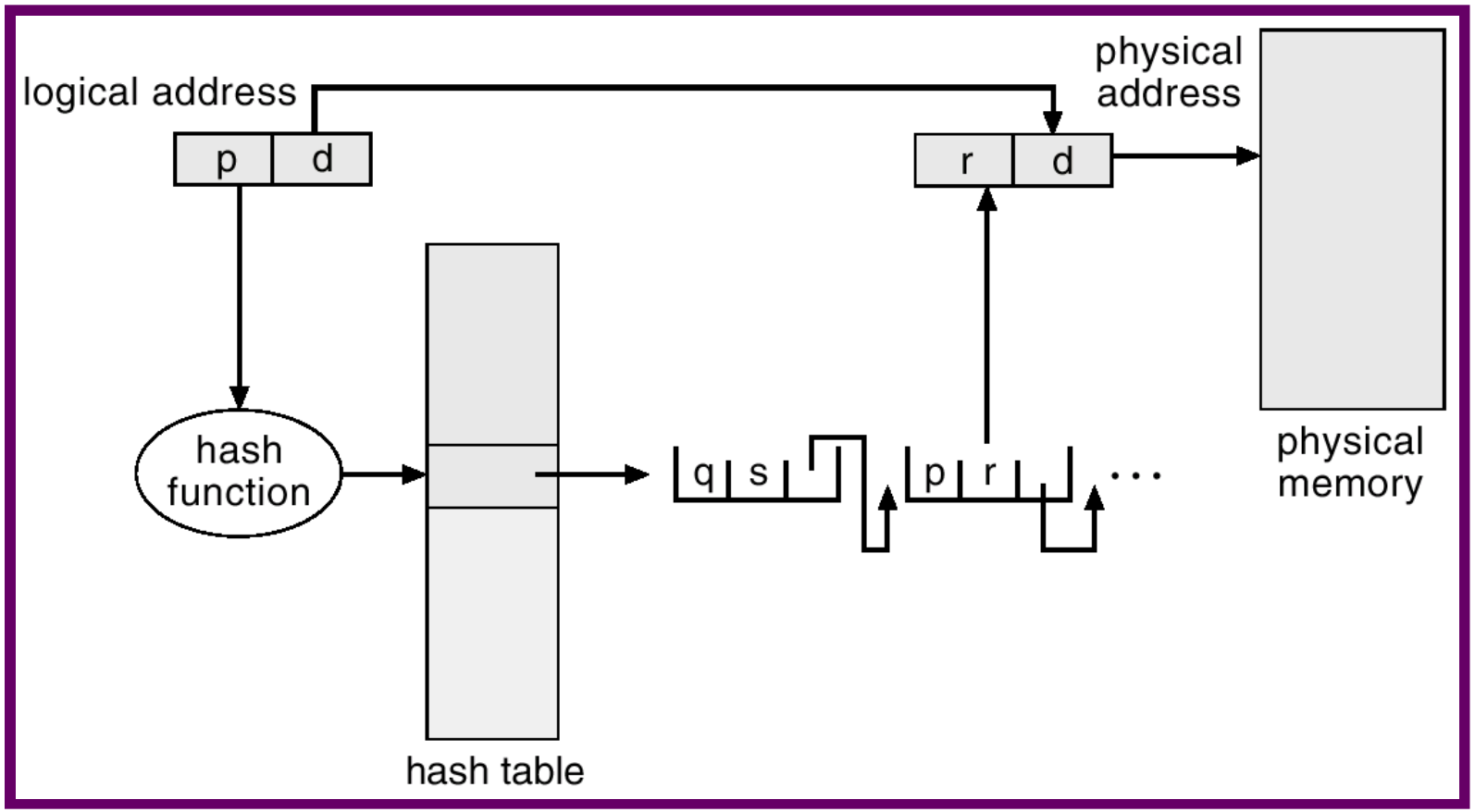


Tabela de Página c/ Hash



Tabelas de páginas invertidas

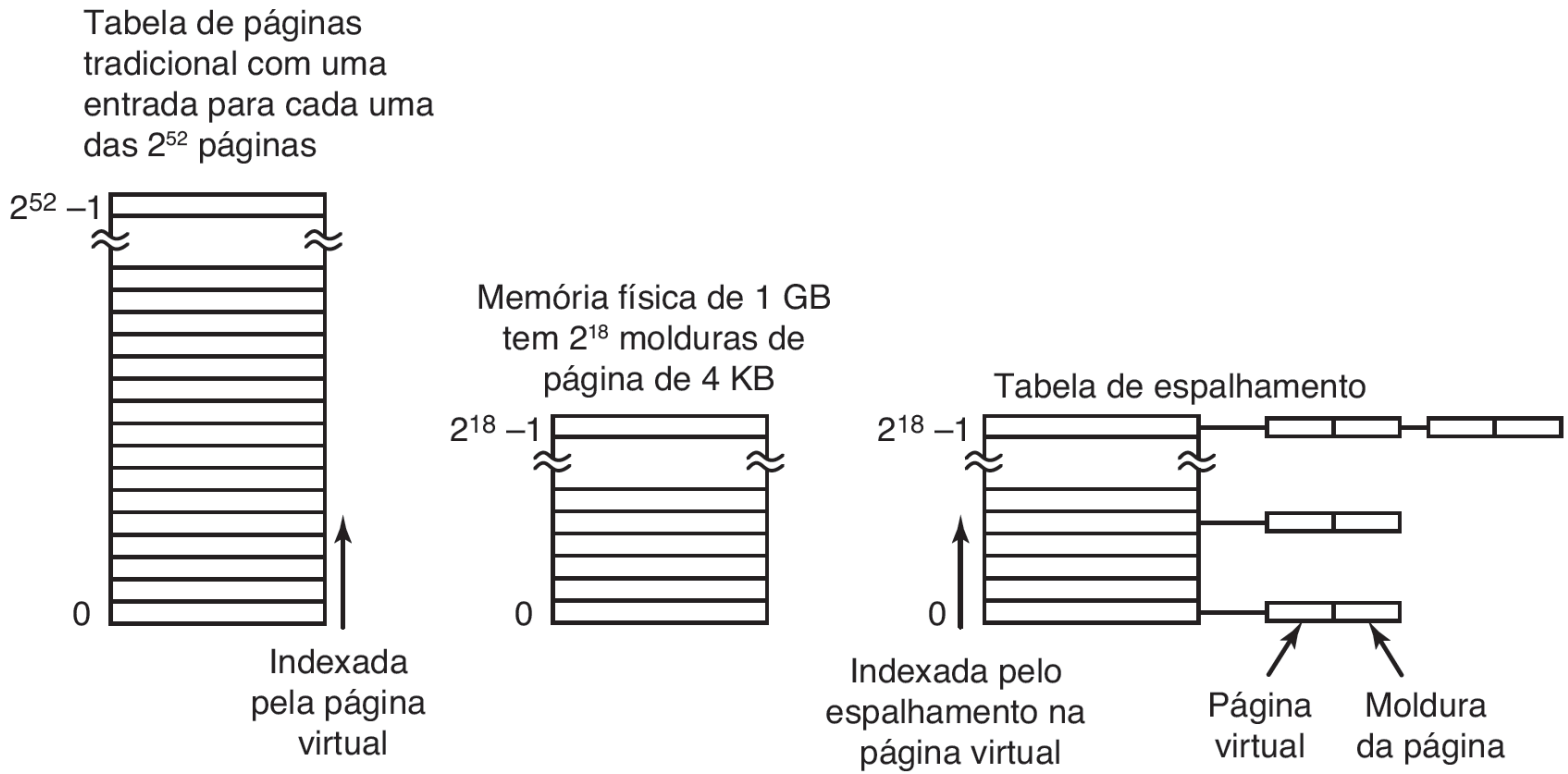


Figura 3.13 Comparação entre uma tabela de páginas tradicional e uma tabela de páginas invertidas.

Outros Problemas…

- Nós assumimos que sempre existe frames livres na memória física.
- E se:
  - Não existe frames livres.
  - Todos os frames estão ocupados.

- É preciso um algoritmo de troca de páginas!
  - Continua…