Aula 15 – TLB, Alocação e Troca de Páginas

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

6 de novembro de 2014

Memória Associativa (TLB)

- O gerenciamento da TLB pode ser:
  - Por software (Spark, MIPS)
    - As entradas da TLB são carregadas pelo SO
    - Em caso de *page fault* na TLB, a MMU faz uma interrupção ao SO
    - O SO busca a página na tabela de páginas, destitui uma entrada da TLB (que está no hardware), insere aí a nova página e reinicializa a instrução interrompida
    - Usa a memória (mais lento)
    - As páginas contendo a própria tabela de páginas podem não estar na TLB, quando do processamento de uma TLB miss – Causará novas TLB misses durante o processo

Memória Associativa (TLB)

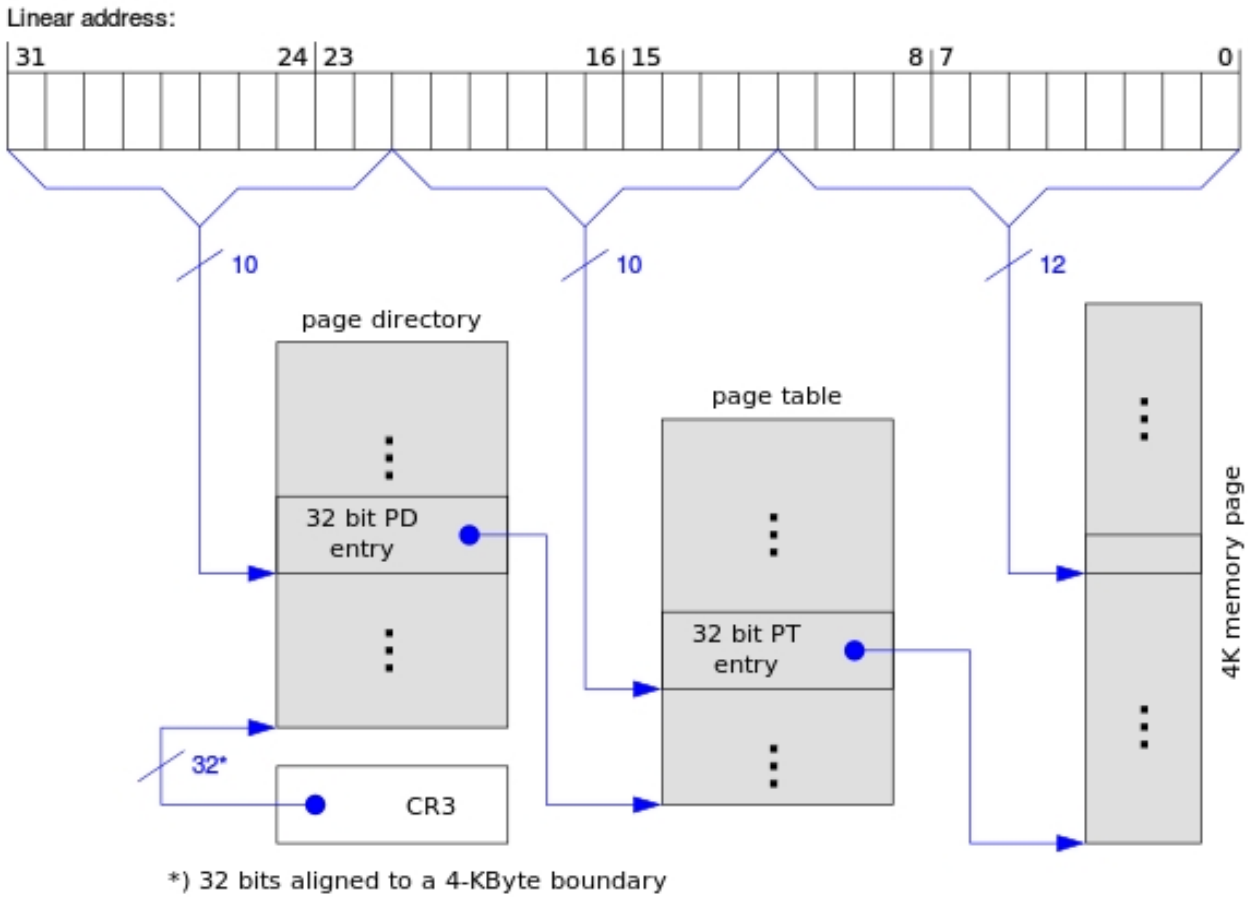
- Gerenciamento por software: tipos de ausências
  - Soft miss
    - Quando a página referenciada não está na TLB, mas está na memória física
    - Basta atualizar a TLB
  - Hard miss
    - A página em si não está na memória física (e nem na TLB, naturalmente)
    - Deve-se fazer um acesso ao disco para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

Tabela de Páginas na Memória Principal

- TLBs aceleram a tradução entre endereços virtuais e reais
  - Mas esse é apenas um problema $\rightarrow$ como traduzir rapidamente
- Resta ainda como lidar com grandes espaços de endereços virtuais $\rightarrow$ como organizar a tabela de páginas?
  - Paginação hierárquica (multi-nível)
  - Tabelas de página invertidas
  - Tabelas de página invertidas em hash

Tabela de Páginas Multinível

- Quebre o espaço de endereço lógico em múltiplas tabelas de página. Ex: x86 – 2 níveis



Paginação em Dois Níveis

- A ideia é evitar manter na memória todas as tabelas de página o tempo todo
  - Apenas as necessárias devem estar na memória
  - Ex: Suponha que um programa esteja dividido em:
    - Um espaço para o código (4MB da base da RAM)
    - Um espaço para os dados (4MB seguintes)
    - Um espaço para a pilha (4MB do topo da RAM)
  - Há um buraco gigante vazio na tabela
    - Uma vez que o offset é 12 bits, as páginas têm 4KB
    - Como temos $32 - 12 = 20$ bits para endereçar páginas, temos $2^{20} = 1\text{M}$ possíveis páginas $\rightarrow$ dessas, usou apenas 3K (1K para cada um dos espaços do programa)

Paginação em Dois Níveis

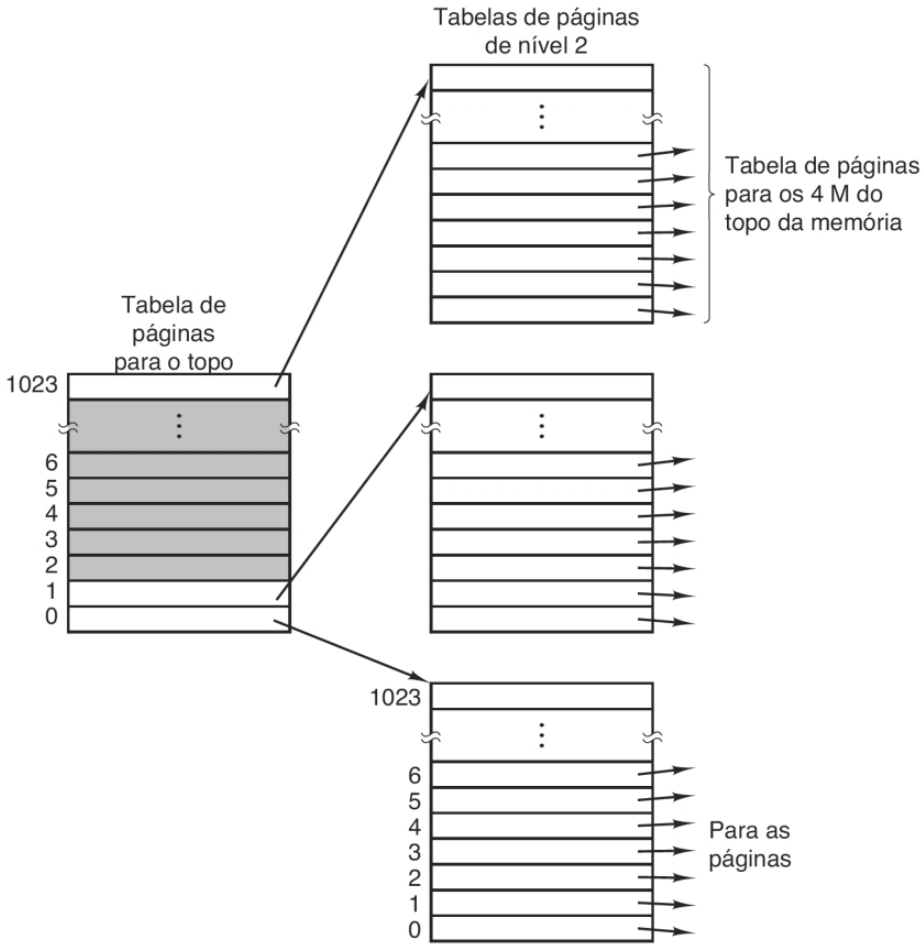
- Lembre que processos mantêm uma tabela completa na memória, quer a usem, quer não
  - Não exatamente verdade – veremos segmentação mais adiante
- Com paginação em mais de um nível, mesmo que a tabela inteira seja enorme, para esse programa precisaríamos apenas manter o suficiente para gerenciar esses 12MB
  - As demais páginas (possivelmente não usadas), não estariam na memória

Paginação em Dois Níveis

Ex: Endereços de 32b e páginas de 4KB

X
X
X
X
7
X
5
X
X
X
3
4
0
6
1
2

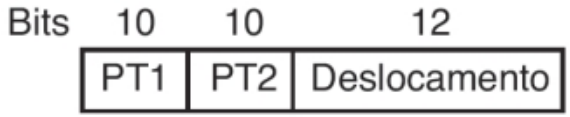
Tabela Única: 20 bits → 1M entradas



Duas Tabelas:
Principal: 10 bits → 1K entradas
Secundárias: Ad hoc

Paginação em Dois Níveis

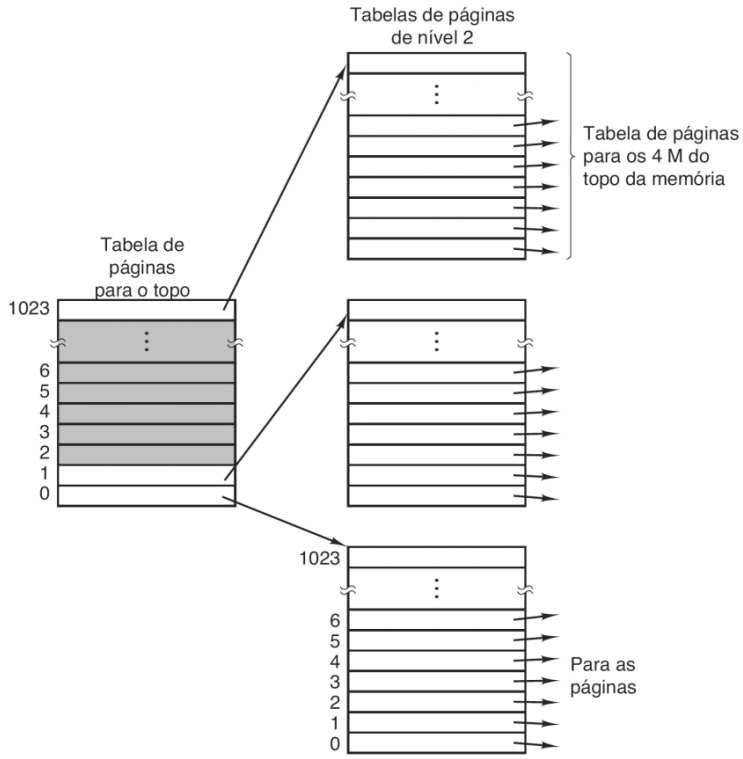
- Endereçamento:
  - Um endereço lógico (em máquinas de 32 bits) é dividido em:
    - Um número de página contendo 20 bits
    - Um deslocamento de página contendo 12 bits



  - Como a tabela de página é paginada, o número de página é dividido ainda em:
    - Um número de página PT1, de 10 bits, que serve de índice para a tabela de página mais externa
    - Um número de página PT2, de 10 bits, que serve de deslocamento da página referenciada dentro da tabela de página mais externa

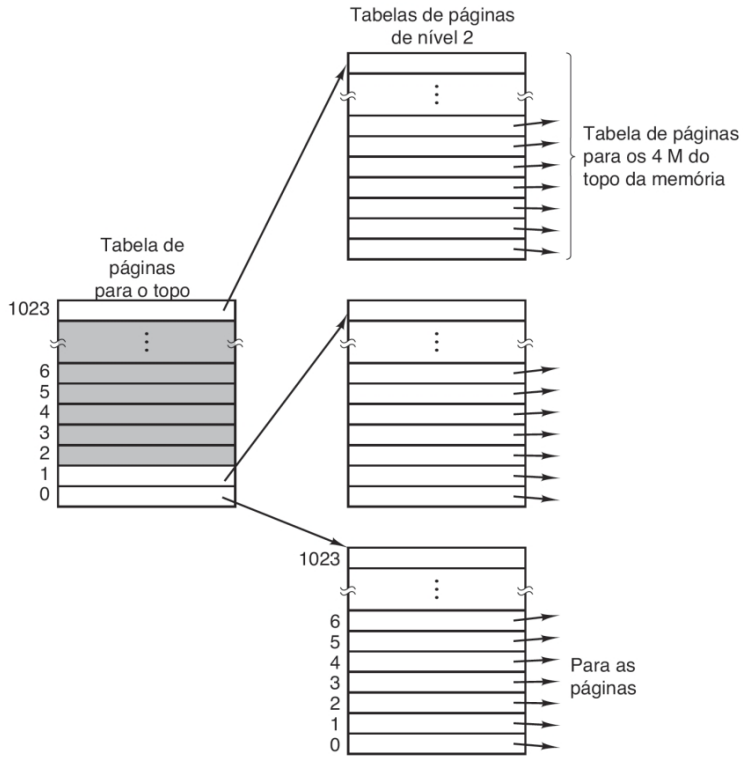
Paginação em Dois Níveis

- Cada página/moldura tem 4KB (12 bits)
  - Como há 1024 entradas na tabela de nivel 2 (por conta dos 10 bits), cada tabela dessas representa 4MB
  - Como cada entrada na tabela de nivel 1 representa uma tabela de nivel 2, cada entrada representa 4MB



Paginação em Dois Níveis

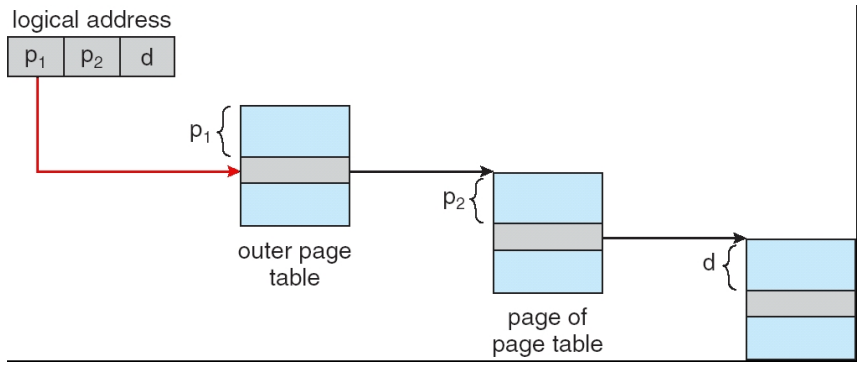
- Cada página/moldura tem 4KB
  - Como há 1024 entradas na de nivel 1 (10 bits), ela representa um todo de 4GB – os 32 bits



- Só precisamos de 4 tabelas na memória
  - A de nivel 1, e as outras 3, para o código (entrada 0), os dados (entrada 1) e a pilha (entrada 1023)

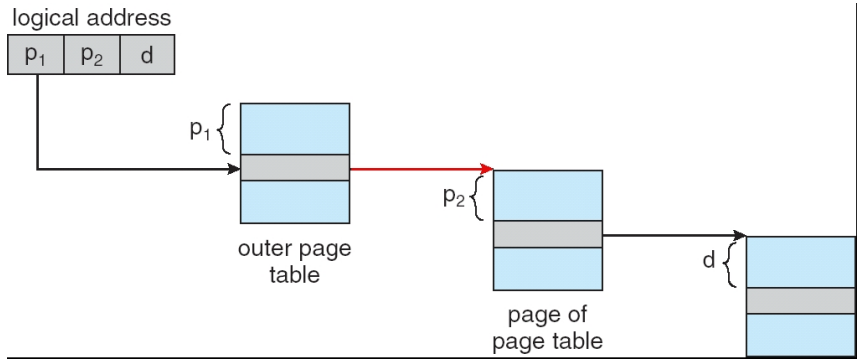
Paginação em Dois Níveis

- Tradução de endereço:
  - Quando chega um endereço virtual, a MMU extrai o campo PT1, usando-o como índice na tabela mais alta (nível 1)
  - O endereço dessa tabela está em um registrador (CR3 no x86)



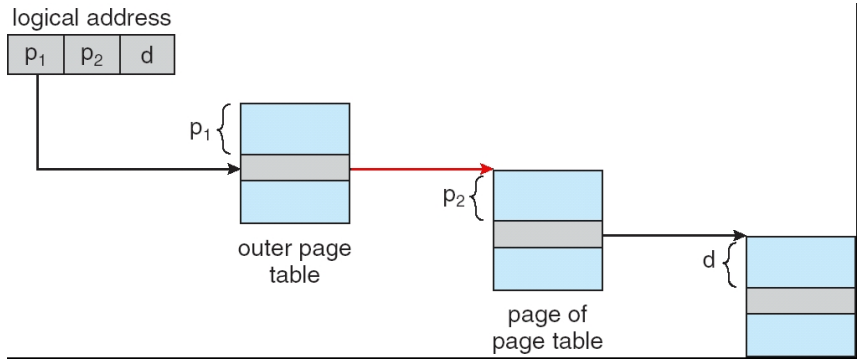
Paginação em Dois Níveis

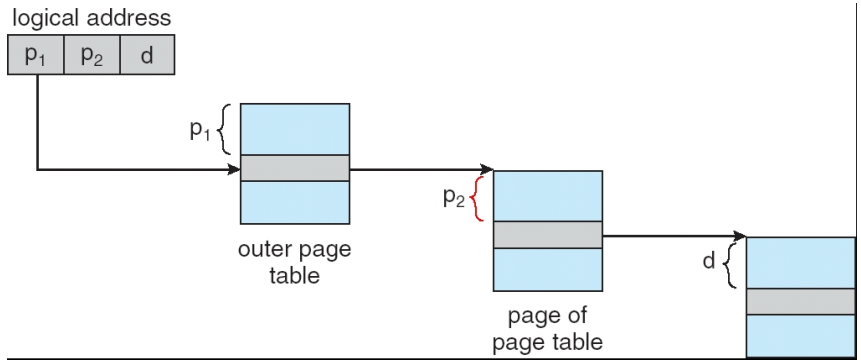
- Tradução de endereço:
  - A entrada nessa posição dá o endereço (ou moldura) da tabela de página no segundo nível
    - Checa o bit de residência



Paginação em Dois Níveis

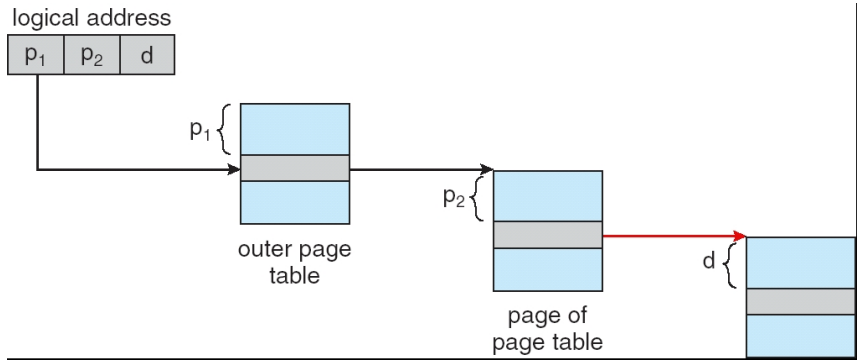
- Tradução de endereço:
  - A entrada nessa posição dá o endereço (ou moldura) da tabela de página no segundo nível
    - Checa o bit de residência
  - O campo PT2 é então usado como índice na tabela de páginas de segundo nível





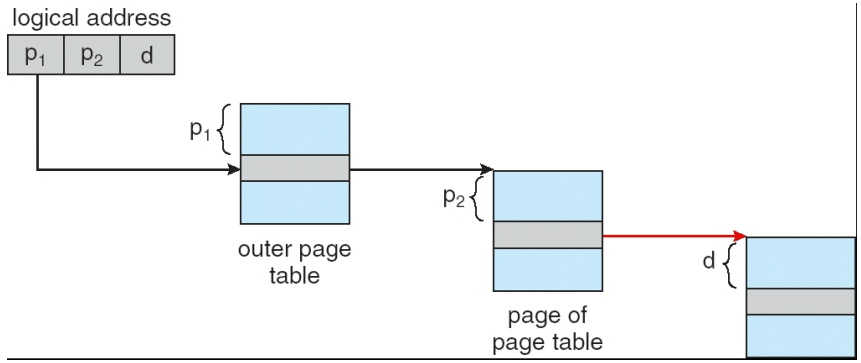
Paginação em Dois Níveis

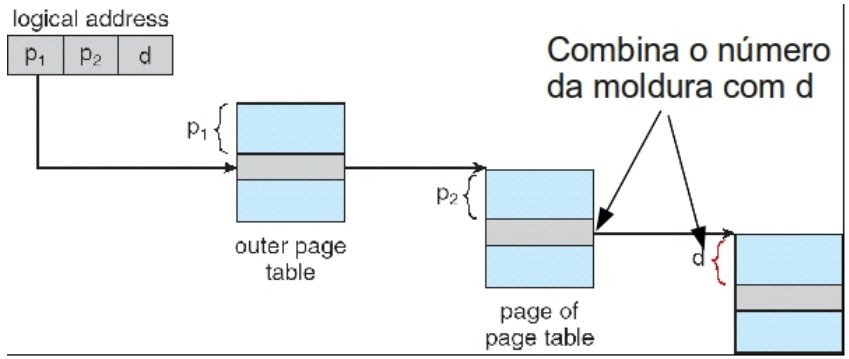
- Tradução de endereço:
  - O número da moldura física é obtido dessa entrada na tabela
    - Checa o bit de residência



Paginação em Dois Níveis

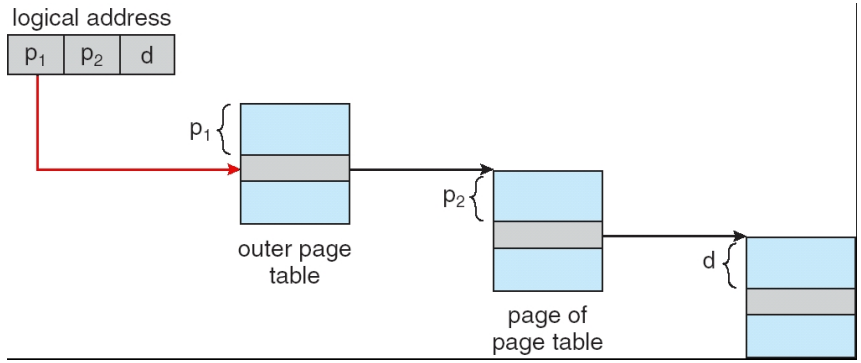
- Tradução de endereço:
  - O número da moldura física é obtido dessa entrada na tabela
    - Checa o bit de residência
  - Finalmente o deslocamento é usado para compor o endereço físico

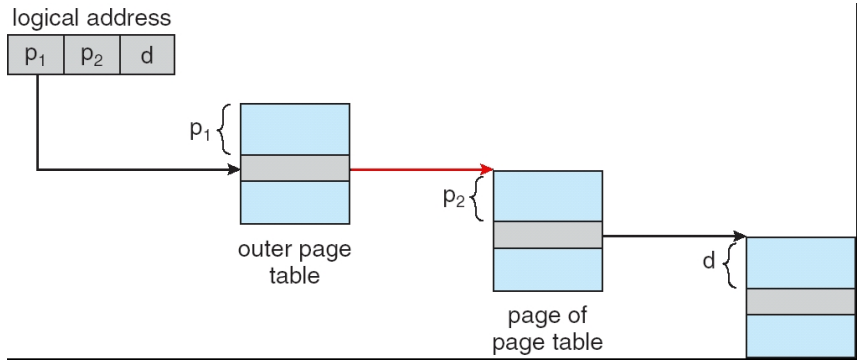




Paginação em Dois Níveis

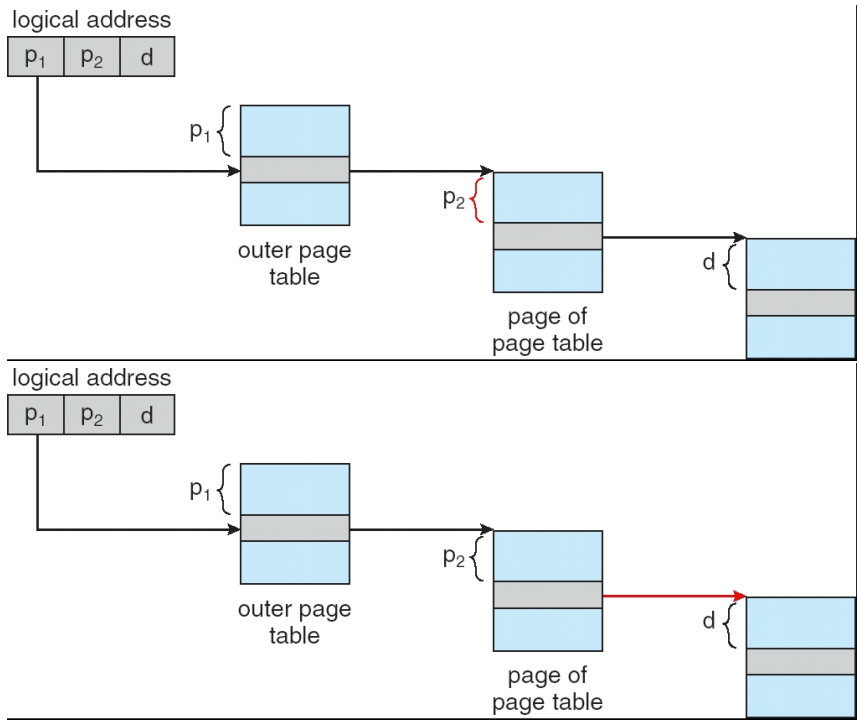
- Ex: 4.206.596 (12.292 do início dos dados – 4MB)
  - 00000000010000000011000000000100
  - PT1 = 1 → entrada 1 da tabela de nível 1
  - O conteúdo, se na memória, contém o endereço da moldura da tabela de nível 2 correspondente





Paginação em Dois Níveis

- Ex: 4.206.596 (12.292 do início dos dados – 4MB)
  - 00000000010000000011000000000100
  - PT2 = 3 → entrada 3 da tabela de nível 2 recém acessada
  - Obtém o número da moldura correspondente



Paginação em Dois Níveis

- Ex: 4.206.596 (12.292 do início dos dados – 4MB)
  - 00000000010000000011000000000100
  - Se a página não estiver na memória:
    - Bit pres/aus = 0, na entrada de nível 2
    - Page fault
  - Se estiver
    - Adiciona ao número da moldura o deslocamento (4) e envia ao barramento

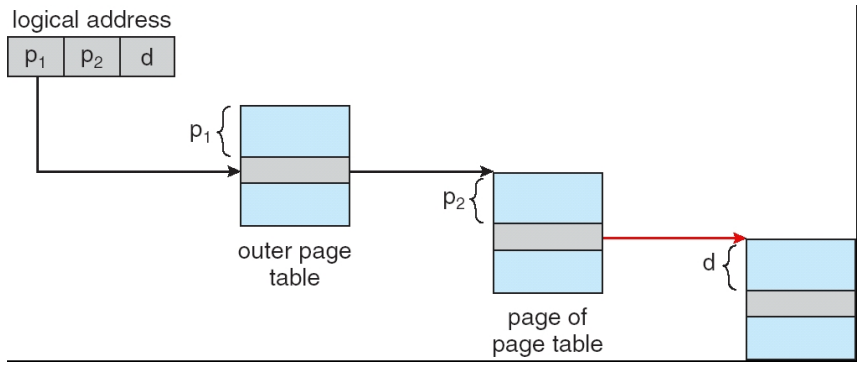


Tabela de Páginas Multinível

- Mais níveis podem existir
  - Dois:

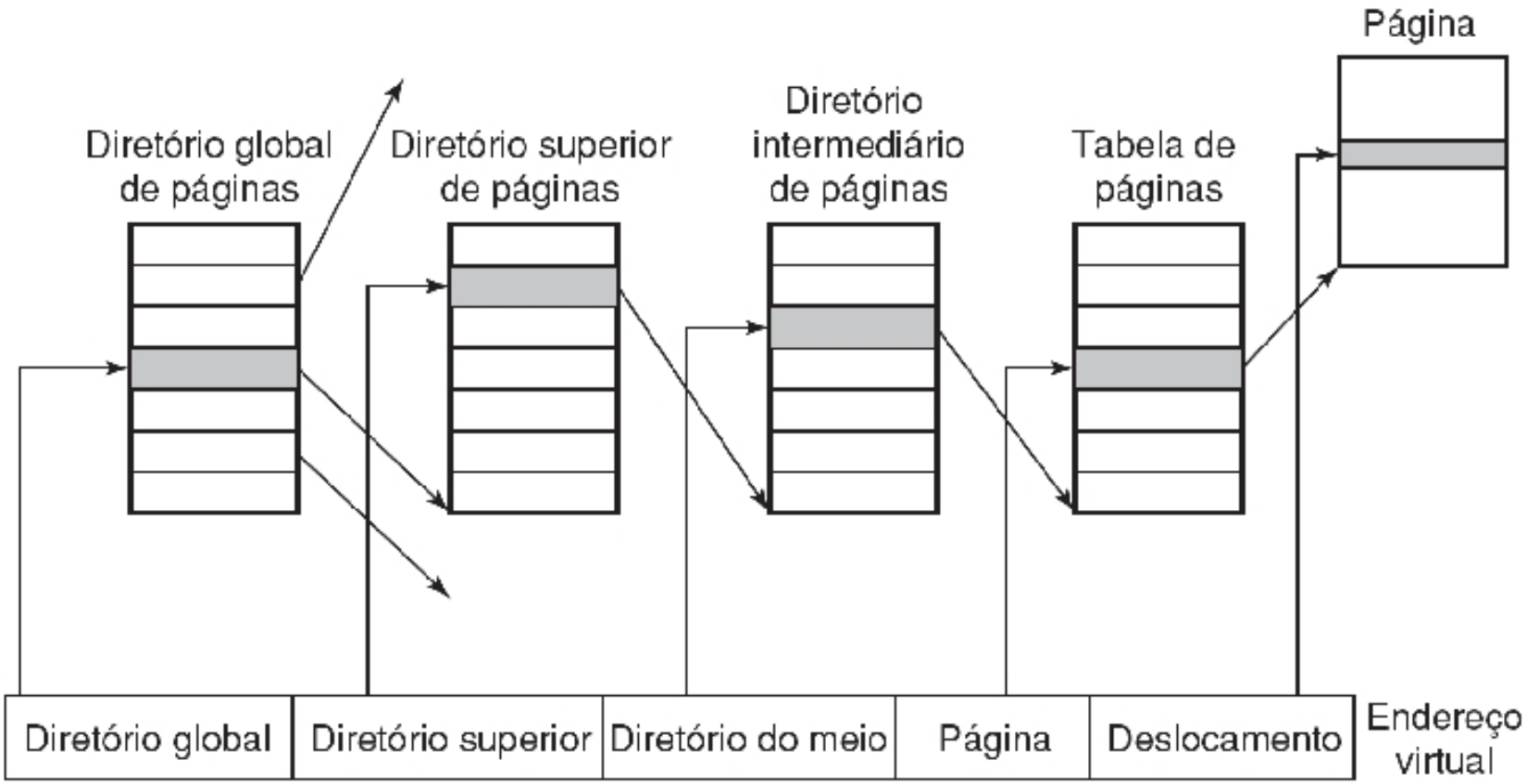
outer page	inner page	offset
p_1	p_2	d
42	10	12

  - Três:

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Tabela de Páginas Multinível

- O Linux utiliza tabelas de páginas de 4 níveis:



Tabelas de Página Invertidas

- Com 32 bits, até 3 níveis vão bem. E com 64?
  - Se a página tiver 4 KB (2^{12}), teremos uma tabela (única) com $2^{64-12} = 2^{52}$ entradas
    - Se cada entrada tiver 8 B, a tabela terá mais de 30.000.000 GB (de fato, 32 PB – peta bytes)
  - Com paginação dupla, 42 bits para a página primária e 10 para a secundária (e 12 para as molduras)
    - $2^{42} \times 8 = 2^{45}$ B (32 TB) só para a primária
  - Com paginação tripla, 32 bits para a página primária e 10 para a secundária e terciária
    - $2^{32} \times 8 = 32GB$ somente para a primária

Tabelas de Páginas Invertidas

- Que fazer? Depende da arquitetura
  - Usar 4 níveis (x86-64 – AMD, depois adotada pela Intel)
    - Com 22-10-10-10-12 bits, temos $2^{22} \times 8 = 32MB$
  - Usar uma tabela de páginas invertida (PowerPC, UltraSPARC e IA-64 – Itanium)
- Tabela que possui uma entrada por moldura física
  - Em vez de uma entrada por página no espaço virtual
  - Indexada pela moldura
  - A entrada registra que página virtual esta localizada na moldura
    - Com informações sobre o processo que possui essa página

Tabelas de Páginas Invertidas

- Poupam muito espaço quando o espaço virtual é muito maior que o físico
- Ex:
  - Endereços virtuais de 64 bits e página de 4 KB
  - Em 1 GB de RAM, a tabela requer tão somente o suficiente para mapear a memória física
    - Nesse caso, 1 GB / 4 KB = 262.144 entradas
  - Independe do tamanho do espaço de endereçamento virtual

Tabelas de Páginas Invertidas

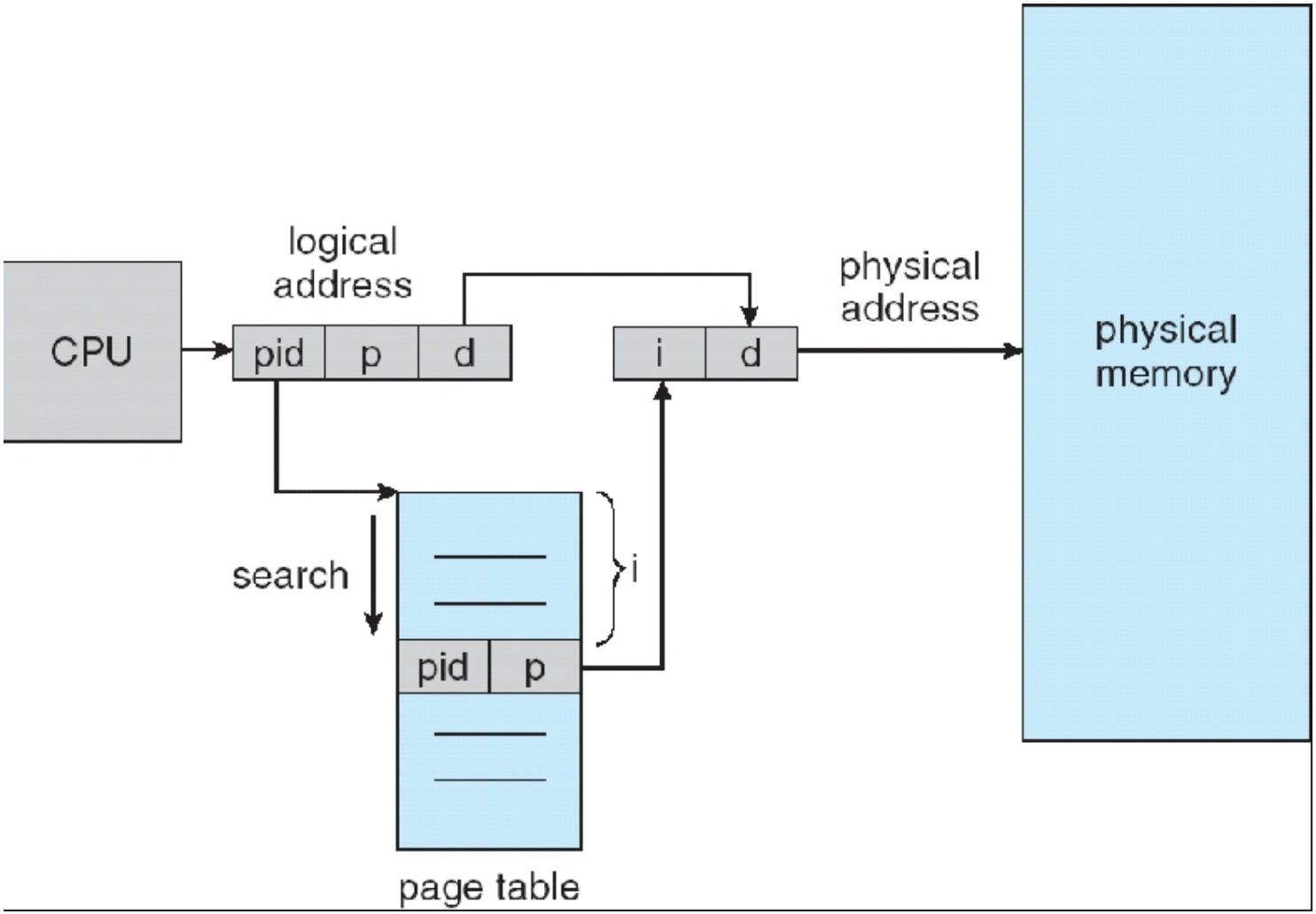
Modelo de única página ou multinível:

- Cada processo tem uma tabela de páginas associada a ele $\rightarrow$ classificação feita pelo endereço virtual

Modelo de página invertida:

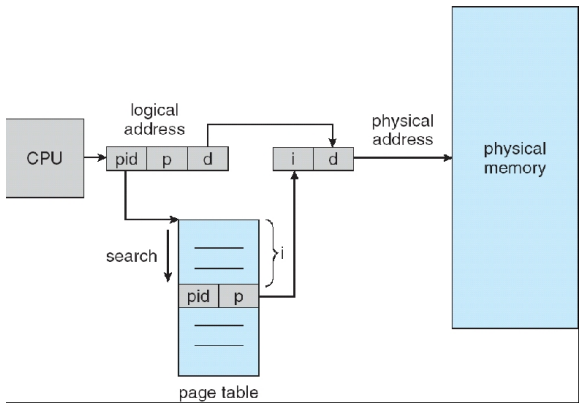
- SO mantém uma única tabela para as molduras de páginas da memória
- Cada entrada consiste no endereço virtual da página armazenada naquela moldura, com informações sobre o processo dono da página virtual

Tabelas de Páginas Invertidas



Tabelas de Páginas Invertidas

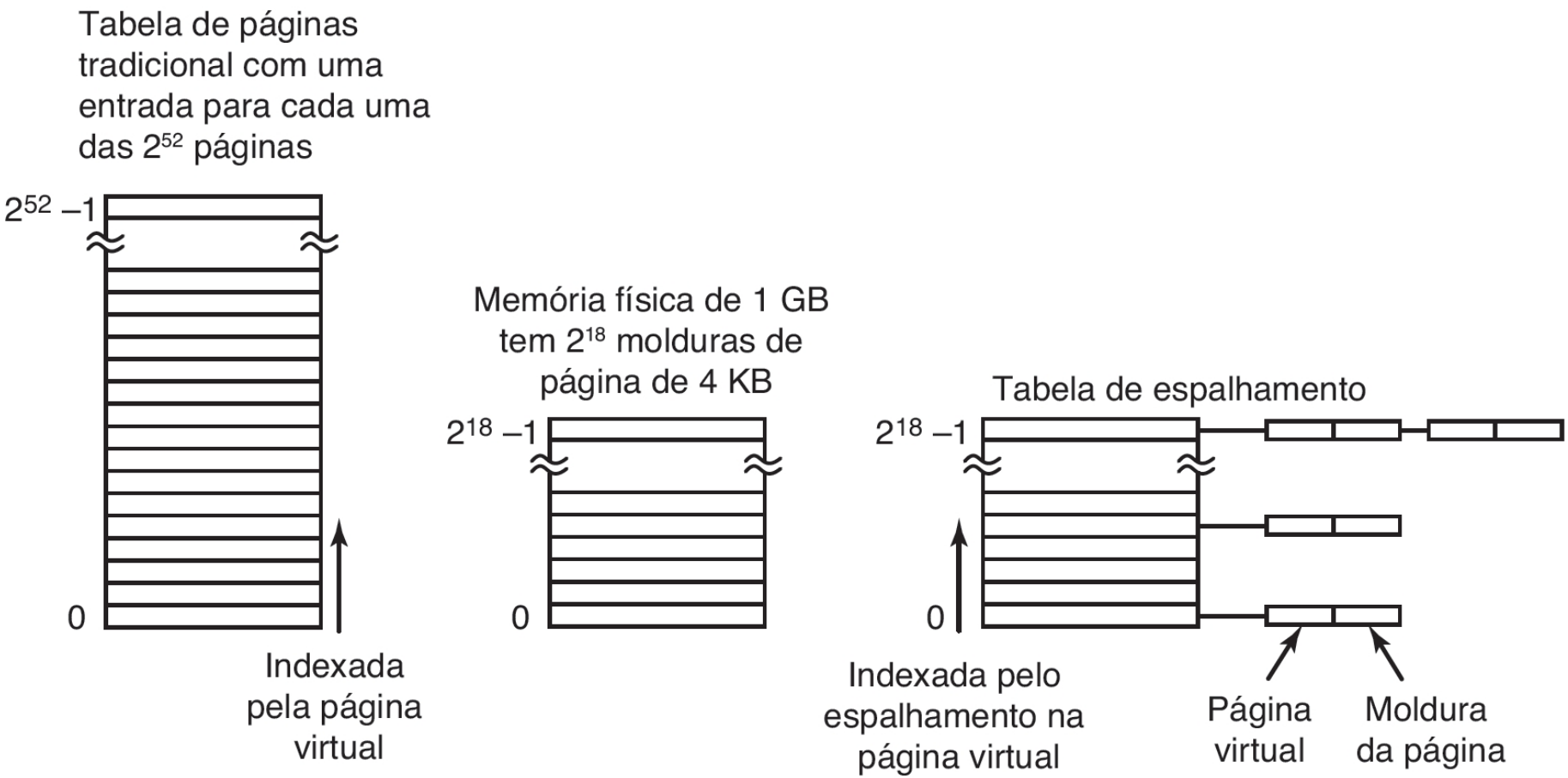
- Problema: Tradução virtual $\rightarrow$ física é muito mais difícil
  - Quando o processo n referencia a página virtual p, o hardware não consegue mais encontrar a página física usando p como índice
    - Deve buscar em toda a tabela invertida por uma entrada (n,p)
    - Se encontrar, gera o endereço físico
    - Faz isso em cada referência à memória, e não somente em *page faults*



Tabelas de Páginas Invertidas

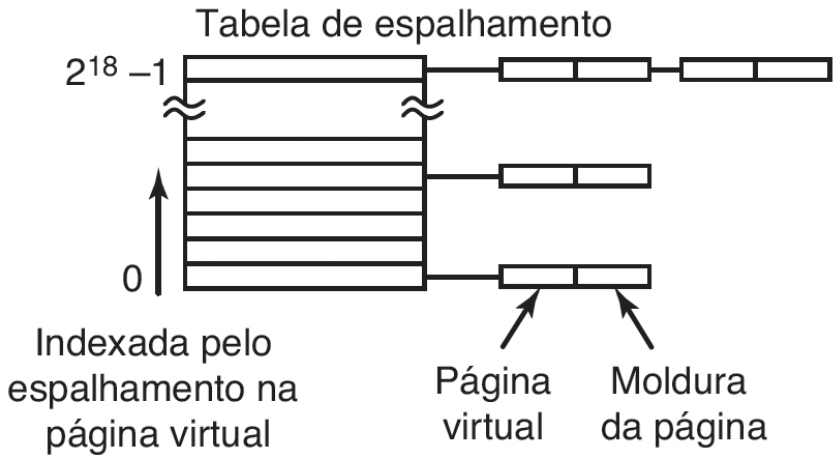
- Que fazer então?
  - Usar a TLB para guardar as páginas mais acessadas
  - Caso a página buscada não esteja na TLB, devemos ainda assim procurar em toda a tabela invertida
- Para auxiliar nessa busca, podemos ter uma tabela de espalhamento (hash) nos endereços virtuais
  - Todas as páginas virtuais atualmente presentes na memória e que tiverem o mesmo valor de hash serão encadeadas juntas

Tabelas de Páginas Invertidas em Hash



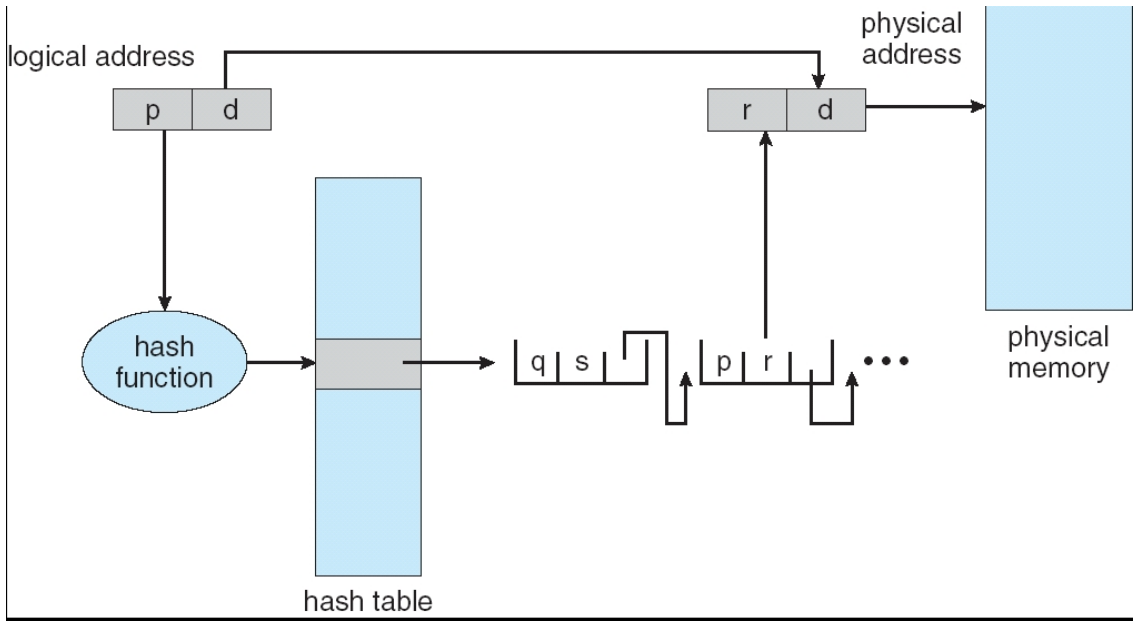
Tabelas de Páginas Invertidas em Hash

- Buscas na tabela invertida são feitas através da tabela de hash, com o valor de hash calculado a partir do endereço virtual
  - Uma vez encontrado o número da moldura (a partir da busca na lista correspondente à página virtual), o par (virtual, físico) é armazenado na TLB



Tabelas de Páginas em Hash

- Funcionamento:
  - O endereço de pagina virtual é usado para cálculo do hash
  - Esse endereço é comparado a cada elemento da lista correspondente
  - Se for encontrado, o campo da moldura é usado no endereço físico
  - Se não, page fault



Tabelas de Páginas Invertidas em Hash

E qual o tamanho de cada lista?

- Se a tabela de hash tiver o mesmo número de entradas que o de molduras, o comprimento médio de cada lista será de somente um par página/moldura
  - Agilizando assim o mapeamento
- Voltamos a ter acesso por meio do endereço virtual, com economia de espaço
- Ao encontrar a moldura, a nova dupla (virtual, física) é inserida na TLB

Que página devemos substituir?

- Política de Substituição Local:
  - Somente páginas dos próprios processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a usada menos recentemente dentre as molduras usadas por A

	Idade
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

idade = instante de último acesso, desde o início do processo

Que página devemos substituir?

- Política de Substituição Local:
  - Somente páginas dos próprios processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a usada menos recentemente dentre as molduras usadas por A

	Idade	
A0	10	A0
A1	7	A1
A2	5	A2
A3	4	A3
A4	6	A4
A5	3	A6
B0	9	B0
B1	4	B1
B2	6	B2
B3	2	B3
B4	5	B4
B5	6	B5
B6	12	B6
C1	3	C1
C2	5	C2
C3	6	C3

idade = instante de último acesso, desde o início do processo

Que página devemos substituir?

- Política de Substituição Local:
  - Somente páginas dos próprios processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a usada menos recentemente dentre as molduras usadas por A
  - Aloca a cada processo fração fixa da memória

	Idade
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

A0
A1
A2
A3
A4
A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

idade = instante de último acesso, desde o início do processo

Que página devemos substituir?

- Política de Substituição Local:
  - Limita cada processo a um número fixo de páginas (fração fixa da memória)
  - Dificuldade: definir quantas páginas cada processo pode utilizar
    - Se o processo usar menos que as previstas $\rightarrow$ desperdício
    - Se usar mais que o previsto (ou galgar mais memória enquanto executa) $\rightarrow$ muitas trocas (ultrapaginação – *thrashing*), mesmo havendo molduras disponíveis
    - Thrashing: diz-se de um processo que gasta mais tempo paginando que executando

Que página devemos substituir?

- Política de Substituição Global:
  - Páginas de todos os processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a mais antiga dentre as molduras disponíveis

	Idade
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

Que página devemos substituir?

- Política de Substituição Global:
  - Páginas de todos os processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a mais antiga dentre as molduras disponíveis

	Idade	
A0	10	A0
A1	7	A1
A2	5	A2
A3	4	A3
A4	6	A4
A5	3	A5
B0	9	B0
B1	4	B1
B2	6	B2
B3	2	A6
B4	5	B4
B5	6	B5
B6	12	B6
C1	3	C1
C2	5	C2
C3	6	C3

Que página devemos substituir?

- Política de Substituição Global:
  - Páginas de todos os processos são utilizadas na troca
    - Ex: page fault no processo A
    - Escolha a mais antiga dentre as molduras disponíveis
  - Processos sem fração fixa da memória

	Idade	
A0	10	A0
A1	7	A1
A2	5	A2
A3	4	A3
A4	6	A4
A5	3	A5
B0	9	B0
B1	4	B1
B2	6	B2
B3	2	A6
B4	5	B4
B5	6	B5
B6	12	B6
C1	3	C1
C2	5	C2
C3	6	C3

Que página devemos substituir?

- Política de Substituição Global:
  - Problema: processos com menor prioridade podem ter um número muito reduzido de páginas, e com isso, acontecem muitas faltas de páginas – *thrashing*
  - Ainda assim, em geral funcionam melhor que os locais

	Idade	
A0	10	A0
A1	7	A1
A2	5	A2
A3	4	A3
A4	6	A4
A5	3	A5
B0	9	B0
B1	4	B1
B2	6	B2
B3	2	A6
B4	5	B4
B5	6	B5
B6	12	B6
C1	3	C1
C2	5	C2
C3	6	C3

Quantas molduras alocar?

- Alocação fixa ou estática:
  - Cada processo tem um número máximo de molduras, definido quando o processo é criado
    - Em um page fault, considera apenas molduras do próprio processo
  - O limite pode ser igual para todos os processos, ou seguir alguma política de distribuição
  - Vantagem: simplicidade
  - Desvantagens:
    - Número muito pequeno de molduras pode causar muita paginação – *thrashing*
    - Número muito grande de molduras causa desperdício de memória principal

Quantas molduras alocar?

- Alocação variável ou dinâmica:
  - Número de molduras alocadas ao processo varia durante sua execução
    - Associada aos algoritmos com política de substituição global
  - Algoritmo: PFF (page faut frequency)
    - Mede a frequência de faltas de página – se alta, aumenta o volume de molduras, se baixa, reduz esse limite
    - Controla tão somente o tamanho do conjunto de alocação
  - Vantagem: processos com elevada taxa de paginação podem ter seu limite de molduras ampliado; e processos com baixa taxa de paginação podem ter seu limite de molduras reduzido
  - Desvantagem: monitoramento constante

Estratégias de Paginação

- Paginação simples:
  - Todas as páginas virtuais do processo são carregadas para a memória principal
  - Assim, sempre todas as páginas são válidas
- Paginação por demanda (Demand Paging):
  - Processos começam com nenhuma página na memória
  - Assim que a CPU tenta executar a primeira instrução, gera um page fault
    - O S.O. traz a página que falta à memória
  - Mais usada hoje

Estratégias de Paginação

- Paginação por demanda (Demand Paging):
  - Apenas as páginas efetivamente acessadas pelo processo são carregadas na memória principal
  - Quais páginas virtuais foram carregadas?
    - Indicado pelo bit de residência (ou referência)
  - Na paginação por demanda as páginas são carregadas na memória somente quando são necessárias

Troca de Páginas

- Relembrando... Quando ocorre um page fault
  - Se todas as molduras estiverem ocupadas, o S.O. escolhe uma página para eliminar – página vítima
    - A escolha pode ser aleatória
    - Será melhor escolher uma página que não seja muito usada
    - Linux: O kernel e o mapa de memória são fixos – suas páginas nunca são excluídas
  - Se essa página foi modificada enquanto estava na memória, deve ser reescrita no disco
  - Se não tiver sido modificada, pode apenas ser sobrescrita

Algoritmos de Troca de Páginas

- Ótimo
- NRU (Not Recently Used)
- FIFO (First-in First-out)
- Segunda Chance
- Relógio
- LRU (Least Recently Used)
- Working set
- WSClock (Working Set Clock)

Algoritmo Ótimo

- Cada página é marcada com o número de instruções que serão executadas antes que a página seja referenciada
  - Retira da memória a página que tem menos chance de ser referenciada (maior número de instruções faltantes)
    - Substitui, dentre as páginas atuais, a que será referenciada por último
  - Adia ao máximo a próxima falta de página
- Praticamente impossível de se saber – Impraticável
- Usado em simulações para comparação com outros algoritmos
  - Executando-se o programa e guardando suas referências às páginas, para então testar na segunda execução

NRU – Not Recently Used

- Classifica as páginas com frames atualmente na memória:
  - Usa os bits de status (associados a cada página) da tabela de páginas → R(eferenciada) e M(odificada)
    - Classe 0 (00) → não referenciada, não modificada
    - Classe 1 (01) → não referenciada, modificada
    - Classe 2 (10) → referenciada, não modificada
    - Classe 3 (11) → referenciada, modificada
  - Referenciada (bit 1) → lida ou escrita
  - Modificada (bit 1) → escrita

NRU – Not Recently Used

- R e M são atualizados a cada referência à memória
  - Armazenados em cada entrada da tabela de página
  - Seu valor é determinado pelo hardware
- Quando um processo é iniciado, ambos R e M são 0 para todas suas páginas
  - Periodicamente, o bit R é limpo para diferenciar as páginas que não foram referenciadas recentemente
    - Operação feita a cada interrupção de relógio (em geral 20ms)

NRU – Not Recently Used

- O bit M não é limpo, pois o S.O. precisa saber se deve escrever a página no disco
- Quando ocorre uma page fault:
  - Remove aleatoriamente uma página, escolhendo dentre as classes mais baixas $\rightarrow$ bits 00, 01, 10, 11
  - Dá preferência a quem não foi recentemente usada e não precisa voltar ao disco
- Vantagens:
  - Fácil de entender, eficiente para implementar e fornece bom desempenho

Referências Adicionais

- http://pages.cs.wisc.edu/~solomon/cs537/html/paging.html
- http://en.wikipedia.org/wiki/Page_replacement_algorithm
- http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html