

Aula 9

Professores:

Felipe M. G. França
Valmir C. Barbosa

Conteúdo:

Gerenciamento de memória

- Tabelas de páginas
- TLB - Translation Lookaside Buffer
- Algoritmos de substituição de página

Tabela de páginas

- ➡ É usada para mapear endereços virtuais em físicos, através do mapeamento de páginas virtuais em molduras de páginas:
 - O número da página da parte superior do endereço virtual é convertido no número da moldura anexado ao deslocamento.
- ➡ A tabela pode ser vista como uma função matemática, que mapeia as páginas virtuais em molduras de página:
 - Se o bit ausente/presente indicar que a página virtual está na memória, a tabela mapeia esta página na moldura de página.
 - Caso contrário, uma falha de página é gerada, e:
 - O sistema operacional é chamado, escolhe uma moldura de página, e copia a página desta moldura no disco.
 - A página é copiada do disco para a moldura, e com isso, a página será mapeada nesta moldura.

Tabela de páginas

- ➡ Cada um dos processos do sistema possui a sua própria tabela de páginas, e com isso:
 - Cada um possui o seu espaço de endereçamento virtual.
 - Um processo não poderá acessar o espaço do outro processo, pois este não estará mapeado na sua tabela.
- ➡ Precisamos tratar de dois fatores importantes em relação ao uso das tabelas de páginas:
 - O tamanho da tabela de páginas, que depende do tamanho do espaço de endereçamento virtual, e do tamanho da página:
 - A tabela para um endereço virtual de 32 bits possui cerca de 1 milhão de entradas, se o tamanho da página for de 4Kb.
 - O tempo necessário para se mapear uma página, pois este mapeamento deve ser feito a cada referência à memória:
 - Ao executar uma instrução, o processador acessa pelo menos uma vez a memória, para obter esta instrução.

Tabela de páginas



Uma possibilidade é a da tabela estar dentro da MMU:

- A tabela é composta por registradores do hardware, que são muito mais rápidos do que a memória.
- Quando um processo for executar no processador, o sistema copia a tabela de páginas do processo para a tabela da MMU.
- Os mapeamentos dos endereços serão feitos através desta tabela, sem usar a memória:
 - O mapeamento agora será bem mais rápido.
 - O problema é o custo de colocar a tabela na MMU, e o tempo necessário para a troca de contexto dos processos.



Quando a tabela de páginas está dentro da MMU, esta tabela possui as páginas mapeadas do processo que está atualmente executando no processador.



Tabela de páginas



Uma possibilidade é a da tabela estar dentro da MMU:

- A tabela é composta por registradores do hardware, que são muito mais rápidos do que a memória.
- Quando um processo for executar no processador, o sistema copia a tabela de páginas do processo para a tabela da MMU.
- Os mapeamentos dos endereços serão feitos através desta tabela, sem usar a memória:
 - O mapeamento agora será bem mais rápido.
 - O problema é o custo de colocar a tabela na MMU, e o tempo necessário para a troca de contexto dos processos.



Quando o escalonador escolhe um novo processo, por exemplo, o n, a tabela do processo que está atualmente executando, por exemplo, o 1, deve ser salva na memória, e a tabela do processo n, escolhido pelo escalonador, deve ser carregada na tabela de páginas da MMU. Logo, a troca de contexto dos processos é muito demorada



Tabela de páginas

→ Uma outra possibilidade é a de manter a tabela na memória:

- O processador possui somente um registrador que aponta para o início da tabela de páginas na memória.
- Quando um processo for executar no processador, o sistema altera o registrador para o endereço da tabela do processo.
- Os mapeamentos deverão acessar a memória, e com isso:
 - Ao trocar os contextos dos processos, precisaremos alterar somente o valor do registrador.
 - O mapeamento será mais lento, pois deveremos acessar a tabela na memória a cada referência à memória.



Se a tabela de páginas está somente na memória do computador, existirá na MMU somente um ponteiro para o início da tabela de páginas do processo que está atualmente executando no processador.



Tabela de páginas

→ Uma outra possibilidade é a de manter a tabela na memória:

- O processador possui somente um registrador que aponta para o início da tabela de páginas na memória.
- Quando um processo for executar no processador, o sistema altera o registrador para o endereço da tabela do processo.
- Os mapeamentos deverão acessar a memória, e com isso:
 - Ao trocar os contextos dos processos, precisaremos alterar somente o valor do registrador.
 - O mapeamento será mais lento, pois deveremos acessar a tabela na memória a cada referência à memória.



Agora, a troca de contexto será bem mais simples: quando o escalonador escolher um novo processo para executar, por exemplo, o n , bastará alterarmos o ponteiro da MMU para o endereço inicial da tabela do processo recém-escolhido. O problema neste caso é que cada acesso à memória irá gerar pelo menos mais um acesso à memória.



Tabela de páginas



O conceito de tabela de páginas multinível objetiva:

- Reduzir o tamanho da tabela de páginas na memória, dividindo esta tabela, e mantendo somente as partes usadas na memória.
- A parte do endereço virtual que define a página é dividida em dois ou mais campos, que dependem do número de níveis:



Divisão do campo número de página em n campos adicionais PT_1, PT_2, \dots, PT_n , onde n é o número de níveis. O campo PT_i define a moldura da página referenciada, se $i = n$, ou a entrada na próxima tabela no nível $i+1$, se $i < n$. Como podemos ver pela divisão do campo número de página, cada entrada da tabela do nível i representa um espaço de memória virtual igual a $PT_{i+1}x\dots xPT_n x Deslocamento$.



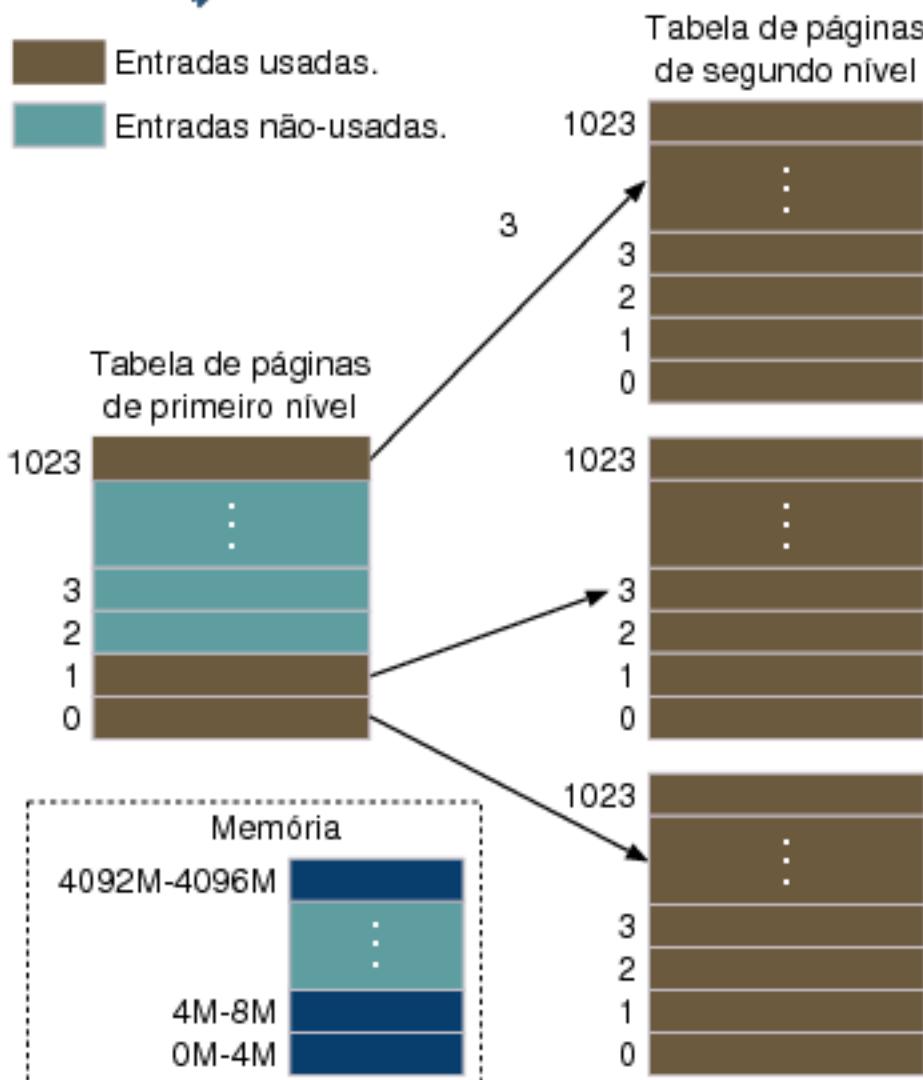
O número total de páginas ainda dependerá somente do tamanho da página, dado pelo campo deslocamento.

Tabela de páginas



Exemplo de uma tabela de páginas com dois níveis:

- Entradas usadas.
- Entradas não-usadas.



Se o computador possui endereços virtuais de 32 bits, e tamanho de página de 4Kb, então os campos número de página e deslocamento terão, respectivamente, 10 bits e 12 bits. Agora, se o campo número de página for dividido em dois campos iguais de 10 bits, então cada entrada da tabela de páginas do primeiro nível representará um espaço de 4M de memória.

Ao lado, vemos as tabelas do programa, sendo que o código ocupa os 4M iniciais da memória, os dados ocupam os 4M seguintes, e a pilha ocupa os últimos 4M da memória.

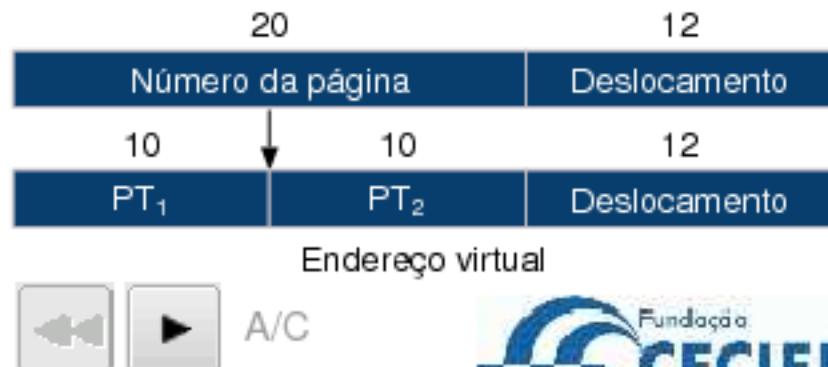
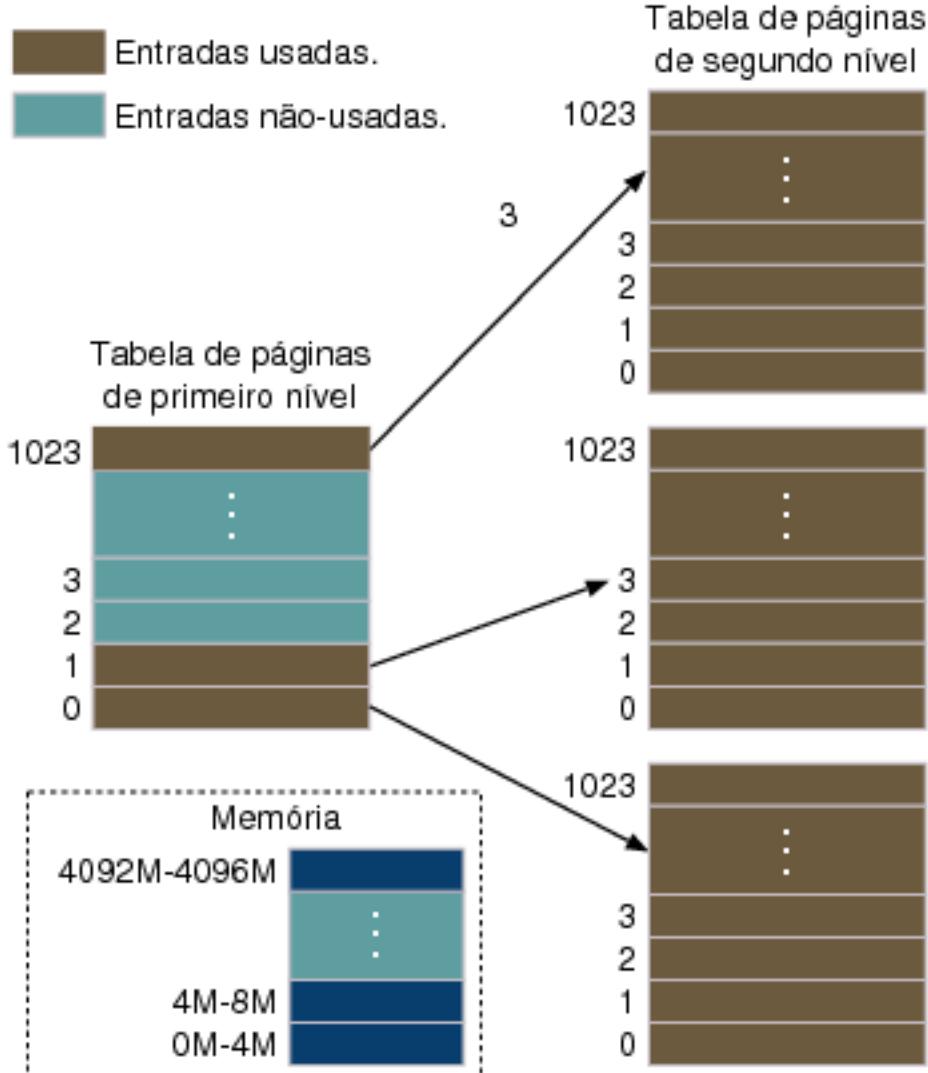


Tabela de páginas



Exemplo de uma tabela de páginas com dois níveis:



Suponha que o endereço virtual 00403004 em hexadecimal (4206596 em decimal), que está a 12292 bytes do início da área de dados, foi acessado. Então, como o endereço virtual possui 32 bits, e as páginas tem 4K, o número da página será 1027 (que começa no endereço 4206592), e o deslocamento será de 4 bytes. Como o valor do campo número de página é 1027, os valores dos campos PT_1 e PT_2 serão, respectivamente, 1 e 3.

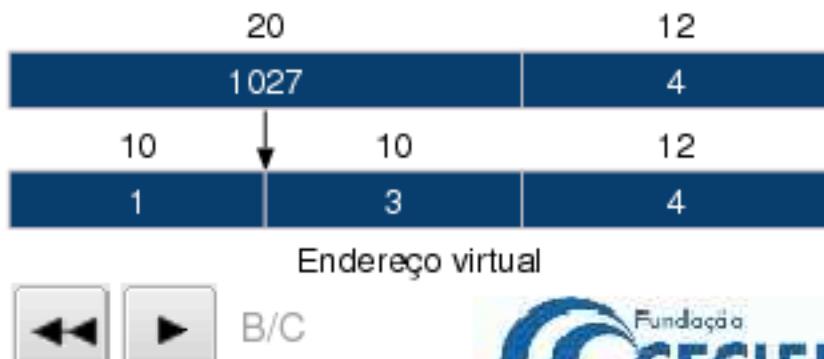
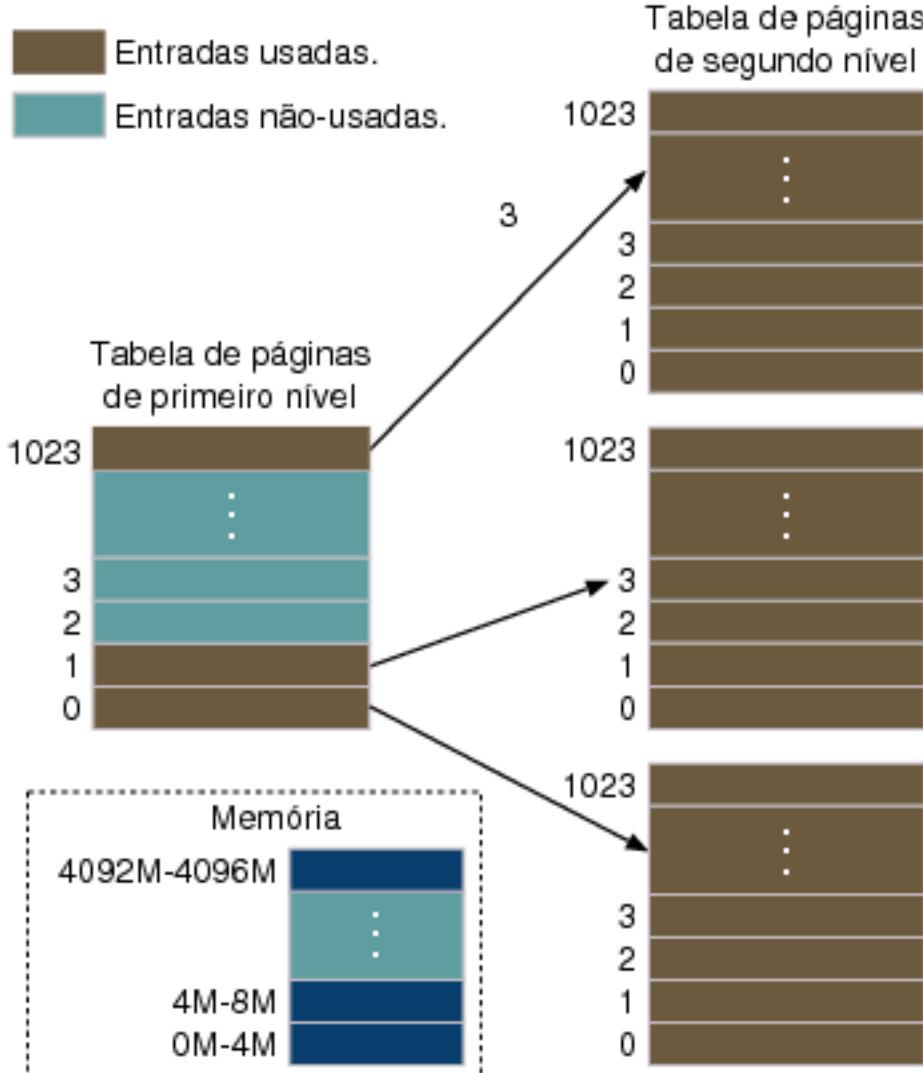


Tabela de páginas



Exemplo de uma tabela de páginas com dois níveis:



Usando o valor de PT_1 , acessamos a entrada 1 da tabela de primeiro nível, que mapeia os endereços de 4M a 8M, para obter o ponteiro para a tabela de segundo nível. Agora, usando o valor 3 de PT_2 , acessamos a entrada 3 desta tabela, e obtemos a moldura de página se o bit ausente/presente for igual a 1. Caso contrário, uma falha de página será gerada, e o sistema operacional pode trazer a página acessada para a memória. O número da moldura, junto com o valor 4 do deslocamento, será usado para formar o endereço físico enviado para a memória.

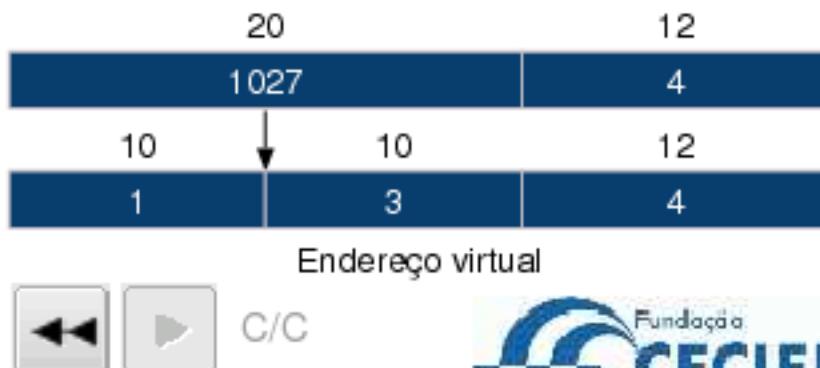
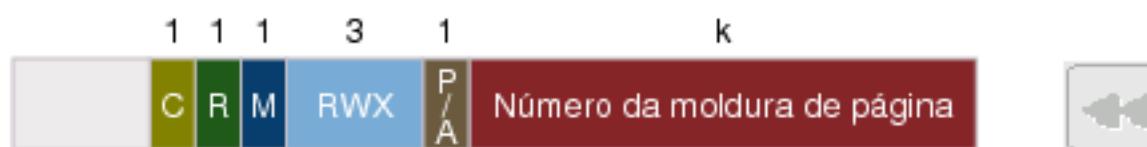


Tabela de páginas

- ➡ Nesta organização, o bit ausente/presente também é usado para garantir a proteção de acesso:
 - As entradas das tabelas associadas às áreas da memória não alocadas ao processo possuem o valor 0 para este bit.
 - Quando uma falha de página for gerada, o sistema pode ver se o processo pode acessar a área da memória na página.
- ➡ O conteúdo de uma entrada da tabela de páginas depende do hardware, mas, em geral, possui o seguinte formato:



Exemplo de um possível formato para cada uma das entradas da tabela de páginas. O número total de molduras de páginas, neste exemplo, é de 2^k . O número de bits da entrada gastos para cada um dos campos é dado acima do campo. Pressione o mouse sobre cada um dos campos para ver uma descrição resumida do seu significado.

TLB

- ➡ Como vimos, a execução das instruções em sistemas baseados na paginação é mais lenta, devido aos acessos à tabela de páginas.
- ➡ Os programas, em geral, tendem a acessar somente uma pequena parte de suas páginas virtuais em um dado instante de tempo.
- ➡ Com isso, foi desenvolvido um dispositivo de hardware, chamado de TLB (Translation Lookaside Buffer), que:
 - Pode armazenar uma pequena quantidade de entradas da tabela de páginas, como as entradas das páginas mais usadas.
 - Possibilita que o mapeamento de uma página para a moldura de página seja feita sem acessar a tabela de páginas.
- ➡ Em geral, a TLB é uma memória associativa interna à MMU, com acesso bem mais rápido do que ao da memória do computador.

TLB



Cada entrada da TLB está associada a uma página, e:

- Possui as mesmas informações sobre esta página dadas na sua entrada da tabela de páginas.
- A única informação adicional é o bit **Válida**, que se for igual a 1, indica que a entrada da TLB é válida.

Válida	Página virtual	Modificada	Proteção	Moldura de página
1	140	1	RW	31
1	20	0	RX	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	RX	50
1	21	0	RX	45
1	860	1	RW	14
1	861	1	RW	75

A tabela dada ao lado é de um programa em que o código, que está atualmente sendo executado, está nas páginas de 19 à 21. Os dados do programa estão nas páginas 129, 130, e 140, e a pilha da execução está nas páginas 860 e 861.

Exemplo de uma TLB com 8 entradas. Cada entrada possui, além do bit Válida, os campos da entrada da tabela de páginas necessários ao se referenciar uma página. Se a página está na TLB, a sua informação é obtida da TLB, ao invés da tabela de páginas.

TLB



Gerenciamento da TLB em hardware, pela MMU:

- Se a página estiver na TLB, as informações sobre a página são obtidas da entrada da TLB associada a esta página.
- Se a página não estiver na TLB, então a MMU:
 - Obtém as informações da página na tabela de páginas.
 - Copia estas informações para uma das entradas da TLB.

Válida	Página virtual	Modificada	Proteção	Moldura de página
1	140	1	RW	31
1	20	0	RX	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	RX	50
1	21	0	RX	45
1	860	1	RW	14
1	861	1	RW	75



A/C

Vamos ver como a MMU usa a TLB para acelerar a conversão dos endereços virtuais nos endereços físicos, para o exemplo que vimos anteriormente, o do programa em que o código que está sendo executado está nas páginas de 19 à 21, os dados nas páginas 129, 130, e 140, e a pilha nas páginas 860 e 861.

TLB



Gerenciamento da TLB em hardware, pela MMU:

- Se a página estiver na TLB, as informações sobre a página são obtidas da entrada da TLB associada a esta página.
- Se a página não estiver na TLB, então a MMU:
 - Obtém as informações da página na tabela de páginas.
 - Copia estas informações para uma das entradas da TLB.

Válida	Página virtual	Modificada	Proteção	Moldura de página
1	140	1	RW	31
1	20	0	RX	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	RX	50
1	21	0	RX	45
1	860	1	RW	14
1	861	1	RW	75



B/C

Se o programa acessar a página 21 ao executar uma instrução, as informações da página serão obtidas na TLB, sem acessar a memória. A MMU deve verificar se o acesso à página é válido, pois, por exemplo, não podemos escrever na página 21.

TLB



Gerenciamento da TLB em hardware, pela MMU:

- Se a página estiver na TLB, as informações sobre a página são obtidas da entrada da TLB associada a esta página.
- Se a página não estiver na TLB, então a MMU:
 - Obtém as informações da página na tabela de páginas.
 - Copia estas informações para uma das entradas da TLB.

Válida	Página virtual	Modificada	Proteção	Moldura de página
1	140	1	RW	31
1	20	0	RX	38
1	130	1	RW	29
1	129	1	RW	62
1	22	0	RX	7
1	21	0	RX	45
1	860	1	RW	14
1	861	1	RW	75



C/C

Se agora o programa executa uma instrução na página 22, a página deverá ser lida da tabela de páginas, pois não está na TLB, e alguma entrada, por exemplo, a da página 19, deverá ser descartada, e o seu bit Modificada deverá ser salvo na tabela de páginas. A página 22 será copiada nesta entrada, evitando que futuras referências a esta página acessem a memória.

TLB



Gerenciamento da TLB pelo sistema operacional:

- A MMU gera uma interrupção, **chamada de uma falha de TLB**, quando não acha a página na TLB:
 - O sistema escolhe a entrada da TLB que será descartada, e copia as informações da página nesta entrada.
- Esta cópia deve ser rápida, pois as falhas de TLB ocorrem com maior freqüência do que as falhas de página.
- Reduz a complexidade do projeto da MMU.



O gerenciamento pelo sistema pode ser acelerado, se:

- O número de entradas da TLB for suficientemente grande, pois reduziremos as falhas na TLB.
- O sistema tentar prever as páginas que serão usadas, para carregá-las na TLB antes que uma falha de TLB seja gerada.
- O sistema manter uma cache de TLB, cuja página sempre está na TLB, e usá-la sempre que uma falha de TLB ocorrer.

Algoritmos de substituição de página

- ➡ Quando uma falha de página ocorrer, o sistema operacional deve:
 - Escolher uma página para ser retirada da memória.
 - Copiar esta página para o disco, caso esta tenha sido alterada.
- ➡ Poderíamos escolher aleatoriamente a página a ser salva no disco, mas esta escolha não garante um bom desempenho ao sistema:
 - Se escolhermos uma página muito usada, esta irá gerar novas falhas de página.
- ➡ Vamos estudar os seguintes algoritmos de substituição de páginas:
 - Algoritmo ótimo.
 - Algoritmo não recentemente utilizada (NRU).
 - Algoritmo primeira a entrar, primeira a sair (FIFO).
 - Algoritmo de segunda chance.
 - Algoritmo do relógio.
 - Algoritmo menos recentemente utilizada (LRU).

Algoritmo ótimo

- É o melhor algoritmo para a escolha da página a ser substituída:
 - Rotula cada página na memória com o número de instruções que serão executadas antes desta ser acessada.
 - Quando ocorre uma falha de página, o algoritmo escolhe a página que será referenciada por último (a com o maior rótulo).
- Infelizmente o algoritmo é impossível de ser implementado:
 - O sistema operacional não tem como saber quantas instruções serão executadas antes de uma página ser referenciada.
- A utilidade do algoritmo vem do fato de que este pode ser usado para avaliar a eficiência de outros algoritmos de substituição:
 - Executamos um processo monitorando as falhas de página, e o número de instruções executadas até acessarmos cada página.
 - O desempenho do algoritmo pode ser medido se rodarmos uma simulação usando as informações obtidas anteriormente.

Algoritmo NRU

- Usa os bits referenciada (R) e modificada (M) da entrada da tabela de páginas, para dividir as páginas em quatro classes:
 - **Classe 0:** páginas não-referenciadas e não-modificadas.
 - **Classe 1:** páginas não-referenciadas e modificadas.
 - **Classe 2:** páginas referenciadas e não-modificadas.
 - **Classe 3:** páginas referenciadas e modificadas.
- O algoritmo, chamado de NRU (Not Recently Used), escolhe uma página da classe não vazia com o menor número, e:
 - Quando o processo é iniciado, todos os bits R e M são limpos.
 - Quando a página é acessada, o bit R é ligado, assim como o bit M, se o processo alterar o conteúdo da página.
 - Periodicamente, o sistema deve limpar todos os bits R.

Página	M	R	Moldura	Classe
20	0	0	38	0
130	1	0	29	1
129	1	1	62	3
19	0	0	50	0
21	0	1	45	2

Exemplo de um processo que usa as páginas 19, 20, 21, 129, 130, e 140, que não está na memória. De acordo com as referências e as alterações das páginas pelo processo, as páginas foram divididas do seguinte modo: classe 0 - páginas 19 e 20; classe 1 - página 130, classe 2 - página 21, e classe 3 - página 129.



A/B

Algoritmo NRU

- Usa os bits referenciada (R) e modificada (M) da entrada da tabela de páginas, para dividir as páginas em quatro classes:
 - **Classe 0:** páginas não-referenciadas e não-modificadas.
 - **Classe 1:** páginas não-referenciadas e modificadas.
 - **Classe 2:** páginas referenciadas e não-modificadas.
 - **Classe 3:** páginas referenciadas e modificadas.
- O algoritmo, chamado de NRU (Not Recently Used), escolhe uma página da classe não vazia com o menor número, e:
 - Quando o processo é iniciado, todos os bits R e M são limpos.
 - Quando a página é acessada, o bit R é ligado, assim como o bit M, se o processo alterar o conteúdo da página.
 - Periodicamente, o sistema deve limpar todos os bits R.

Página	M	R	Moldura	Classe
20	0	0	38	0
130	1	0	29	1
129	1	1	62	3
140	0	1	50	2
21	0	1	45	2

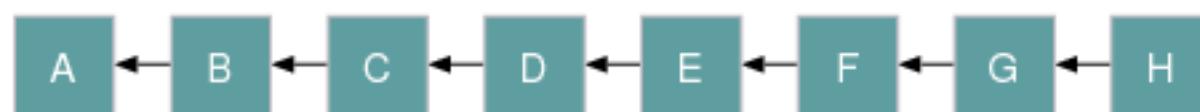


B/B

Suponha agora que o processo acessou a página 140, que não está mapeada. Isso irá gerar uma falha de página, e poderemos escolher entre duas páginas: a 19 ou a 20. No exemplo, a página 19 foi a que foi escolhida para ser removida da memória.

Algoritmo FIFO

- ➡ As páginas são organizadas em uma lista, ordenada de acordo com a ordem em que estas páginas foram copiadas na memória:
 - A primeira página da lista é a página que foi copiada a mais tempo na memória.
 - A última página da lista foi a última carregada na memória.
- ➡ Quando ocorre uma falha de página, o algoritmo remove a primeira página da lista, e coloca a página recém-copiada no final da lista.
- ➡ O problema é que a primeira página pode ser a mais intensamente utilizada, e com isso, a sua remoção irá gerar uma nova falha:



Neste exemplo, existem oito páginas atualmente na memória: A, B, C, D, E, F, G, e H. A página A foi a primeira página copiada na memória, e a página H foi a última a ser copiada. Suponha que a página A é a mais referenciada destas oito páginas.



A/C

Algoritmo FIFO

- ➡ As páginas são organizadas em uma lista, ordenada de acordo com a ordem em que estas páginas foram copiadas na memória:
 - A primeira página da lista é a página que foi copiada a mais tempo na memória.
 - A última página da lista foi a última carregada na memória.
- ➡ Quando ocorre uma falha de página, o algoritmo remove a primeira página da lista, e coloca a página recém-copiada no final da lista.
- ➡ O problema é que a primeira página pode ser a mais intensamente utilizada, e com isso, a sua remoção irá gerar uma nova falha:

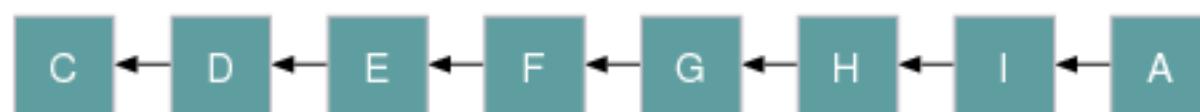


Se uma outra página I, que não está na memória, for referenciada, então uma das páginas deverá ser retirada da memória, e copiada para o disco, se assim for necessário. Se usarmos o algoritmo FIFO, a página que será retirada da memória será a A, mesmo esta sendo a mais referenciada.



Algoritmo FIFO

- ➡ As páginas são organizadas em uma lista, ordenada de acordo com a ordem em que estas páginas foram copiadas na memória:
 - A primeira página da lista é a página que foi copiada a mais tempo na memória.
 - A última página da lista foi a última carregada na memória.
- ➡ Quando ocorre uma falha de página, o algoritmo remove a primeira página da lista, e coloca a página recém-copiada no final da lista.
- ➡ O problema é que a primeira página pode ser a mais intensamente utilizada, e com isso, a sua remoção irá gerar uma nova falha:



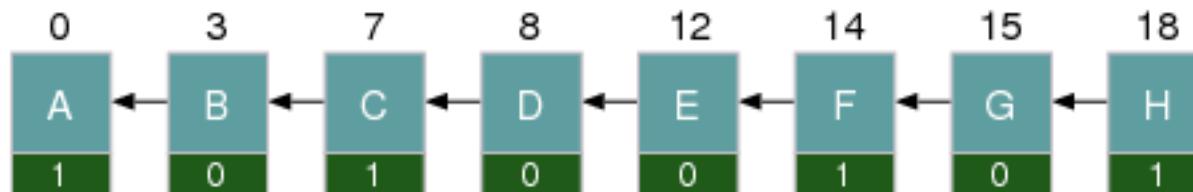
Como a página A é muito referenciada, logo depois da página I ter sido copiada, uma nova falha de página é gerada, para copiar novamente a página A para a memória. A página deletada agora é a B. A ineficiência deste algoritmo é exatamente a deste exemplo, ou seja, a de deletar páginas que vão certamente ser referenciadas mais tarde, o que gerará falhas de página desnecessárias.



C/C

Algoritmo de segunda chance

- Uma versão do algoritmo anterior, baseada no bit referenciada, que tenta evitar a remoção das páginas intensamente utilizadas:
 - Se o bit referenciada da primeira página da lista for 1, então:
 - O bit será limpo (mudado para 0), e a página será movida para o final da lista.
 - O algoritmo continuará verificando a próxima página da lista.
 - Caso contrário, a página será removida da lista, e, assim como antes, a nova página copiada no final da lista.
- Para manter os bits referenciada, o algoritmo usa a mesma idéia do algoritmo NRU que vimos anteriormente.

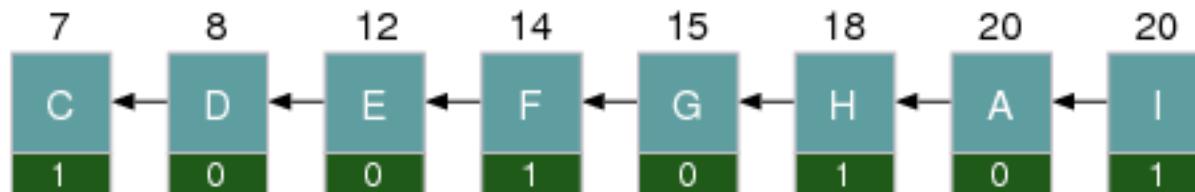


Exemplo dado no algoritmo anterior, com as oito páginas A, B, C, D, E, F, G, e H. Acima das páginas, são dados os tempos em que estas foram copiadas do disco para a memória, e abaixo das páginas, os valores atuais dos bits referenciada. Assim como antes, suponha que a página A é a mais intensamente usada.



Algoritmo de segunda chance

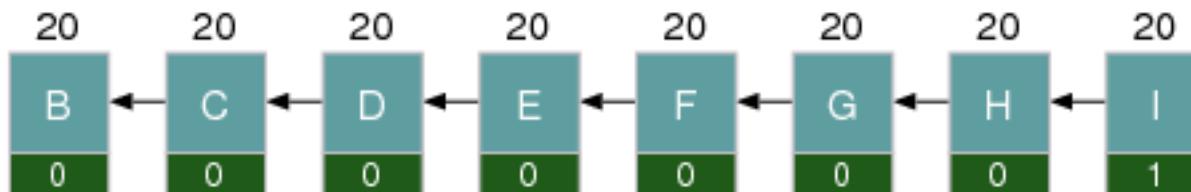
- Uma versão do algoritmo anterior, baseada no bit referenciada, que tenta evitar a remoção das páginas intensamente utilizadas:
 - Se o bit refenciada da primeira página da lista for 1, então:
 - O bit será limpo (mudado para 0), e a página será movida para o final da lista.
 - O algoritmo continuará verificando a próxima página da lista.
 - Caso contrário, a página será removida da lista, e, assim como antes, a nova página copiada no final da lista.
- Para manter os bits referenciada, o algoritmo usa a mesma idéia do algoritmo NRU que vimos anteriormente.



Se usássemos o algoritmo anterior, a página A seria deletada. Mas como o bit referenciada é 1, este bit será limpo, e A será movida para o final da lista. Como o bit referenciada da página B é 0, B será removida da memória e da lista, e a nova página I referenciada será copiada para o final da lista.

Algoritmo de segunda chance

- Uma versão do algoritmo anterior, baseada no bit referenciada, que tenta evitar a remoção das páginas intensamente utilizadas:
 - Se o bit referenciada da primeira página da lista for 1, então:
 - O bit será limpo (mudado para 0), e a página será movida para o final da lista.
 - O algoritmo continuará verificando a próxima página da lista.
 - Caso contrário, a página será removida da lista, e, assim como antes, a nova página copiada no final da lista.
- Para manter os bits referenciada, o algoritmo usa a mesma idéia do algoritmo NRU que vimos anteriormente.



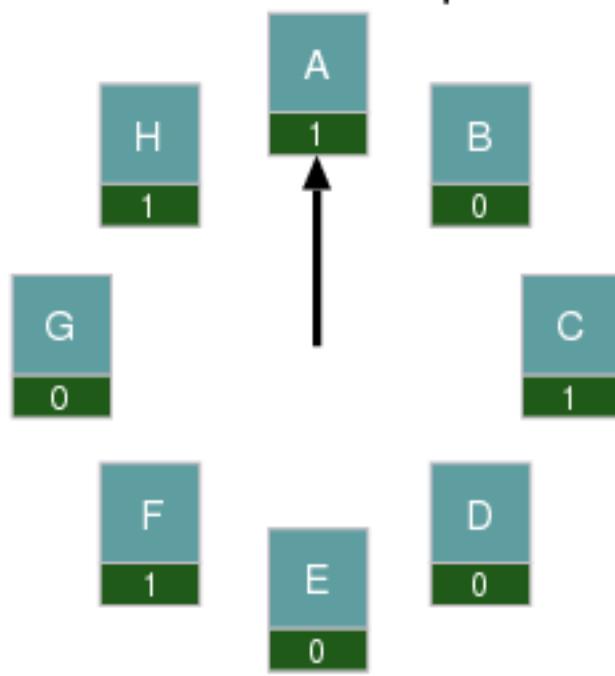
Se, ao acessarmos a página I, todas as páginas foram referenciadas, então simplesmente iremos movê-las, uma a uma, para o final da lista, para finalmente depois escolher a página A para ser removida. Com isso, o algoritmo sempre termina, e encontra uma página para ser removida da memória.

Algoritmo do relógio



Uma implementação eficiente do algoritmo anterior:

- Mantém uma lista circular com as páginas, com um ponteiro apontando para próxima página a ser substituída.
- Se ocorrer uma falha de página, então, para esta página:
 - Se o bit referenciada é 0, então a removemos, copiamos a nova página na sua posição, e atualizamos o ponteiro.
 - Caso contrário, o bit é limpo, o ponteiro é atualizado, e este passo é repetido para a próxima página da lista.



Mesmo exemplo dos dois algoritmos anteriores, em que temos 8 páginas, A, B, C, D, E, F, G, e H, na memória. O ponteiro da lista aponta para a página A, que foi a primeira a ser carregada na memória. Assim como antes, os valores dos bits referenciada são dados abaixo das páginas.

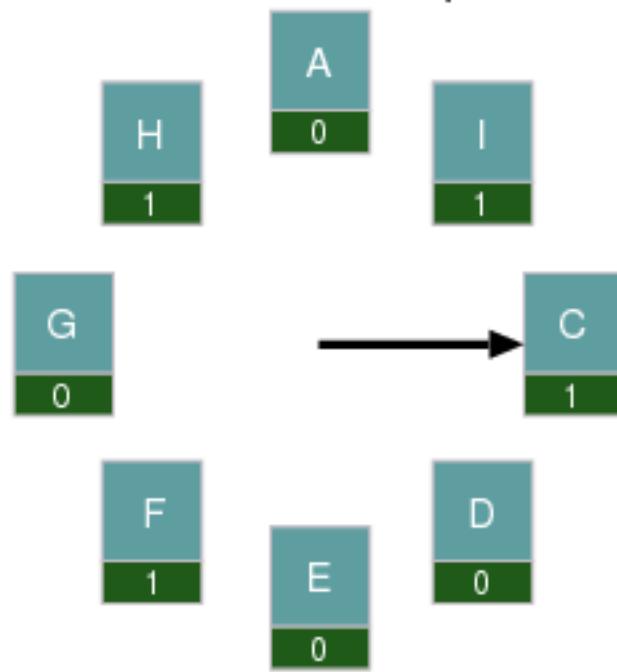


Algoritmo do relógio

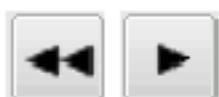


Uma implementação eficiente do algoritmo anterior:

- Mantém uma lista circular com as páginas, com um ponteiro apontando para próxima página a ser substituída.
- Se ocorrer uma falha de página, então, para esta página:
 - Se o bit referenciada é 0, então a removemos, copiamos a nova página na sua posição, e atualizamos o ponteiro.
 - Caso contrário, o bit é limpo, o ponteiro é atualizado, e este passo é repetido para a próxima página da lista.



Suponha que o processo em execução acessou a página I, que não está na memória. O algoritmo verifica primeiramente a página A, apontada pelo ponteiro. Como o bit referenciada é 1, este é limpo, e o ponteiro passa para a próxima página, a B. Agora, como o bit referenciada de B é 0, B é removida, a página I substitui B na lista, e o ponteiro é atualizado, passando a apontar para a página C.



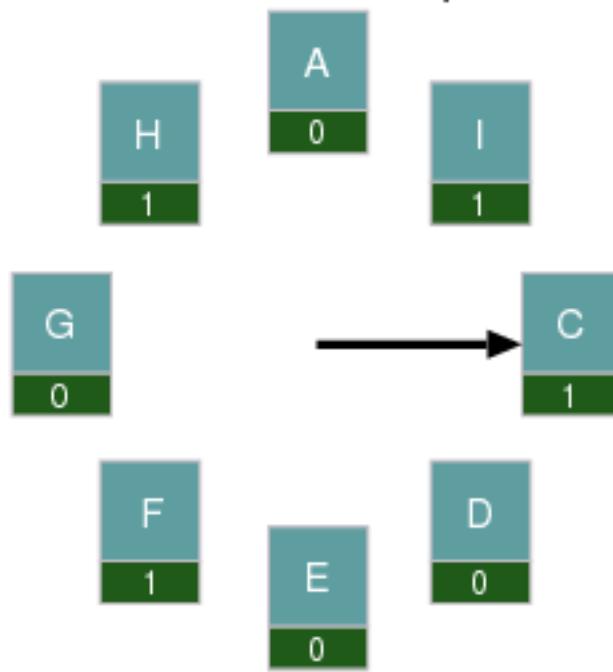
B/C

Algoritmo do relógio



Uma implementação eficiente do algoritmo anterior:

- Mantém uma lista circular com as páginas, com um ponteiro apontando para próxima página a ser substituída.
- Se ocorrer uma falha de página, então, para esta página:
 - Se o bit referenciada é 0, então a removemos, copiamos a nova página na sua posição, e atualizamos o ponteiro.
 - Caso contrário, o bit é limpo, o ponteiro é atualizado, e este passo é repetido para a próxima página da lista.



A grande diferença deste algoritmo e o anterior é que neste alteramos somente o valor do ponteiro, enquanto que no outro era necessário mover as páginas para o final da lista.



Algoritmo LRU

- ➡ O algoritmo é baseado nas seguintes observações:
 - As páginas que foram acessadas pelas últimas instruções provavelmente serão acessadas pelas próximas instruções.
 - As páginas que não foram acessadas por muito tempo provavelmente não serão acessadas por mais algum tempo.
- ➡ Quando ocorre uma falha de página, o algoritmo simplesmente escolhe a página que não foi acessada a mais tempo.
- ➡ O problema do algoritmo é que a sua implementação é complicada:
 - Devemos manter uma lista encadeada composta por todas as páginas na memória, ordenada pelo tempo de acesso.
 - Devemos, a cada referência à memória, procurar a página referenciada na lista, e colocá-la no início da lista.
 - Atualizar a lista a cada referência à memória demanda muito tempo, mesmo se a operação for feita em hardware.

Algoritmo LRU

Um modo de implementar o algoritmo é se o hardware possuir um registrador especial de 64 bits:

- O registrador é incrementado a cada instrução executada.
- Cada entrada da tabela de páginas possui um campo com tamanho de 64 bits para armazenar o contador.
- Quando uma página é referenciada, o campo do contador é atualizado com o valor atual do contador.
- Quando uma falha de página ocorre, a página da tabela com o menor valor para o contador é a escolhida pelo algoritmo.

A/C



Tabela de páginas

	Referenciada	...	Moldura	Contador
0	0	...	31	342
1	1	...	38	2349
2	0	...	29	311
:	:	:	:	:
674	-	...	X	-
:	:	:	:	:
n-1	0	...	45	121

Exemplo de uma tabela de páginas em que cada entrada possui um campo adicional, o **contador**, que contém o valor em que estava o contador do processador quando a página foi acessada pela última vez. No exemplo, o valor do contador do processador é o dado abaixo, obtido após uma instrução acessar a página 1.

Processador
2349 Contador

Algoritmo LRU

Um modo de implementar o algoritmo é se o hardware possuir um registrador especial de 64 bits:

- O registrador é incrementado a cada instrução executada.
- Cada entrada da tabela de páginas possui um campo com tamanho de 64 bits para armazenar o contador.
- Quando uma página é referenciada, o campo do contador é atualizado com o valor atual do contador.
- Quando uma falha de página ocorre, a página da tabela com o menor valor para o contador é a escolhida pelo algoritmo.

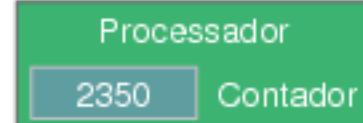
B/C



Tabela de páginas

	Referenciada	...	Moldura	Contador
0	0	...	31	342
1	1	...	38	2349
2	1	...	29	2350
:	:	:	:	:
674	-	...	X	-
:	:	:	:	:
n-1	0	...	45	121

Suponha que uma instrução acessou a página 2. Então, o contador interno do processador será atualizado para 2350, e, na entrada da página 2, o bit referenciada será ligado, e o valor atual do contador do processador será copiado para o contador desta entrada da tabela.



Algoritmo LRU

Um modo de implementar o algoritmo é se o hardware possuir um registrador especial de 64 bits:

- O registrador é incrementado a cada instrução executada.
- Cada entrada da tabela de páginas possui um campo com tamanho de 64 bits para armazenar o contador.
- Quando uma página é referenciada, o campo do contador é atualizado com o valor atual do contador.
- Quando uma falha de página ocorre, a página da tabela com o menor valor para o contador é a escolhida pelo algoritmo.

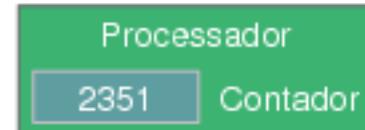
C/C



Tabela de páginas

	Referenciada	...	Moldura	Contador
0	0	...	31	342
1	1	...	38	2349
2	1	...	29	2350
:	:	:	:	:
674	1	...	45	2351
:	:	:	:	:
n-1	-	...	X	-

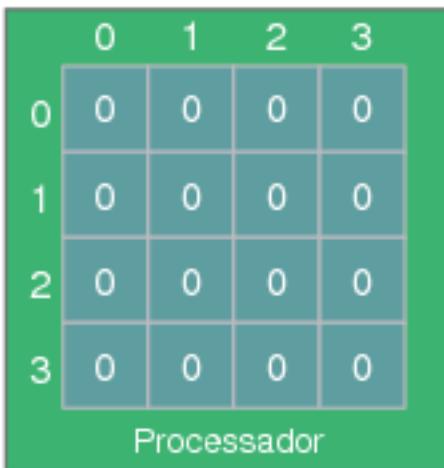
Se agora uma instrução acessar a página 674, que não está mapeada na memória (indicado por um 'X' na entrada da moldura), será gerada uma falha de página. A página com o menor contador, digamos, a n-1, é removida da memória, e a página 674 é copiada para a moldura 45. O contador é atualizado para 2351, e é copiado na entrada 674.



Algoritmo LRU

Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.



A diagram illustrating a memory system. It features a 4x4 grid of squares, each containing a '0'. Above the grid, the columns are labeled 0, 1, 2, and 3. To the left of the grid, the rows are labeled 0, 1, 2, and 3. Below the grid, the word 'Processador' is centered. To the left of the grid, there are two small gray rectangular buttons with arrows: one pointing left and one pointing right.

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Processador

No exemplo temos quatro molduras de páginas, numeradas de 0 até 3. Vamos ver como a matriz varia de acordo com a seguinte ordem da acesso às molduras: 0, 1, 2, 3, 2, 1, 0, 3, 2, e 3. Inicialmente, nenhuma das molduras foi acessada, e com isso, todas as linhas possuem o valor 0000 em binário.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

B/K



	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

Processador

Matriz obtida depois de acessar a moldura 0. As molduras 1, 2, e 3 (que ainda não foram acessadas) são as menos recentemente usadas.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

C/K

◀ ▶

	0	1	2	3
0	0	0	1	1
1	1	0	1	1
2	0	0	0	0
3	0	0	0	0

Processador

Matriz obtida depois de acessar as molduras 0 e 1, nesta ordem. As molduras 2 e 3 (que ainda não foram acessadas) são as menos recentemente usadas.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

D/K

	0	1	2	3
0	0	0	0	1
1	1	0	0	1
2	1	1	0	1
3	0	0	0	0

Processador

Matriz obtida depois de acessar as molduras 0, 1 e 2, nesta ordem. A moldura 3 (que ainda não foi acessada) é a menos recentemente usada.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

E/K

◀ ▶

	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, e 3, nesta ordem. Agora a primeira moldura acessada, a 0, é a menos recentemente usada.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

The diagram illustrates a memory system. At the top, there is a green rectangular area containing a 4x4 grid of squares. The columns are labeled 0, 1, 2, and 3 from left to right. The rows are labeled 0, 1, 2, and 3 from top to bottom. The first row (row 0) contains all zeros (0, 0, 0, 0). The second row (row 1) contains 1, 0, 0, 0. The third row (row 2) contains 1, 1, 0, 1. The fourth row (row 3) contains 1, 1, 0, 0. Below this grid is the word "Processador". To the left of the green area, there is a small grey box with two buttons: one with a double-headed arrow and another with a single right-pointing arrow. Above this box, the letters "F/K" are written.

0	1	2	3
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, e 2, nesta ordem. A moldura 0 ainda é a menos recentemente usada.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

G/K

	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	1	0	0	1
3	1	0	0	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, 2, e 1, nesta ordem. A moldura 0 ainda é a menos recentemente usada.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

H/K

	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, 2, 1, e 0, nesta ordem. Agora, a moldura 3 é a menos recentemente usada, pois as molduras 0, 1, e 2 foram acessadas após a 3.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

I/K

	0	1	2	3
0	0	1	1	0
1	0	0	1	0
2	0	0	0	0
3	1	1	1	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, 2, 1, 0, e 3 nesta ordem. Agora, a moldura 2 é a menos recentemente usada, pois foi acessada antes das molduras 0, 1, e 3.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

J/K

	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	1
3	1	1	0	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, 2, 1, 0, 3, e 2, nesta ordem. Como a moldura 1 foi acessada antes das molduras 0, 2 e 3, esta será agora a menos recentemente usada.

Algoritmo LRU



Um outro algoritmo em hardware, para um sistema com n molduras de página, é o seguinte:

- O hardware possui uma matriz de bits com n linhas e n colunas.
- Quando uma página é referenciada:
 - Os bits da linha da moldura de página associada a página referenciada são todos ligados.
 - Os bits da coluna desta moldura são todos limpos.
- Ao ocorrer uma falha de página, o hardware escolhe a página associada a moldura cuja linha possui o menor valor binário.

K/K

	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	0
3	1	1	1	0

Processador

Matriz obtida depois de acessar as molduras 0, 1, 2, 3, 2, 1, 0, 3, 2, e 3, nesta ordem. A moldura 1 é a menos recentemente usada. Como podemos ver, o valor binário das linhas dão, depois de alguns acessos às molduras, a correta ordem em que estas foram acessadas pela última vez: 3, 2, 0, e 1. Com isso, se ocorrer uma falha de página, o hardware escolherá a página correta, se remover da memória a página 1, cuja linha possui o menor valor binário, igual a 0000.

Algoritmo NFU

- ➡ Caso o hardware não possua suporte para o algoritmo, este deverá ser feito em software.
- ➡ O algoritmo não freqüentemente utilizada (NFU) tenta descobrir a freqüência de uso de cada uma das páginas:
 - O sistema mantém um contador para cada página da tabela, inicializado com o valor 0.
 - Em cada interrupção do temporizador, o sistema, para cada entrada da tabela de páginas, adiciona o bit R ao contador.
 - Quando uma falha de página ocorre, o algoritmo escolhe a página com o menor valor para o contador.
- ➡ O problema do algoritmo é que os contadores medem o número de vezes que a página foi usada:
 - O algoritmo pode escolher uma página em uso porque as outras páginas foram muito mais usadas no passado.

Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0 1 2 3 4 5

0	0	0	0	0	0
---	---	---	---	---	---

0	00000000
---	----------

1	00000000
---	----------

2	00000000
---	----------

3	00000000
---	----------

4	00000000
---	----------

5	00000000
---	----------

Exemplo do algoritmo de idade para 6 páginas, numeradas de 0 até 5. Inicialmente, as páginas ainda não foram acessadas, e os valores dos bits referenciada das páginas são todos iguais a 0. Cada um dos contadores possui 8 bits, e é inicializado com o valor 0.



A/G

Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0	1	2	3	4	5
1	0	1	0	1	1

Vamos ver como serão os valores dos contadores, se ocorrerem os seguintes acessos às páginas, em cinco interrupções consecutivas do temporizador:

0	10000000
---	----------

Após a interrupção 1: as páginas 0, 2, 4, e 5 foram acessadas.

1	00000000
---	----------

Após a interrupção 2: as páginas 0, 1, e 4 foram acessadas.

2	10000000
---	----------

Após a interrupção 3: as páginas 0, 1, 3, e 5 foram acessadas.

3	00000000
---	----------

Após a interrupção 4: as páginas 0 e 4 foram acessadas.

4	10000000
---	----------

Após a interrupção 5: as páginas 1 e 2 foram acessadas.

5	10000000
---	----------



B/G

Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0	1	2	3	4	5
---	---	---	---	---	---

1	1	0	0	1	0
---	---	---	---	---	---

Vamos ver como serão os valores dos contadores, se ocorrerem os seguintes acessos às páginas, em cinco interrupções consecutivas do temporizador:

0 11000000

Após a interrupção 1: as páginas 0, 2, 4, e 5 foram acessadas.

1 10000000

Após a interrupção 2: as páginas 0, 1, e 4 foram acessadas.

2 01000000

Após a interrupção 3: as páginas 0, 1, 3, e 5 foram acessadas.

3 00000000

Após a interrupção 4: as páginas 0 e 4 foram acessadas.

4 11000000

Após a interrupção 5: as páginas 1 e 2 foram acessadas.

5 01000000



Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0	1	2	3	4	5
---	---	---	---	---	---

1	1	0	1	0	1
---	---	---	---	---	---

Vamos ver como serão os valores dos contadores, se ocorrerem os seguintes acessos às páginas, em cinco interrupções consecutivas do temporizador:

0 11100000

Após a interrupção 1: as páginas 0, 2, 4, e 5 foram acessadas.

1 11000000

Após a interrupção 2: as páginas 0, 1, e 4 foram acessadas.

Após a interrupção 3: as páginas 0, 1, 3, e 5 foram acessadas.

2 00100000

Após a interrupção 4: as páginas 0 e 4 foram acessadas.

3 10000000

Após a interrupção 5: as páginas 1 e 2 foram acessadas.

4 01100000

D/G

5 10100000



Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0	1	2	3	4	5
---	---	---	---	---	---

1	0	0	0	1	0
---	---	---	---	---	---

Vamos ver como serão os valores dos contadores, se ocorrerem os seguintes acessos às páginas, em cinco interrupções consecutivas do temporizador:

0 11110000

Após a interrupção 1: as páginas 0, 2, 4, e 5 foram acessadas.

1 01100000

Após a interrupção 2: as páginas 0, 1, e 4 foram acessadas.

2 00010000

Após a interrupção 3: as páginas 0, 1, 3, e 5 foram acessadas.

3 01000000

Após a interrupção 4: as páginas 0 e 4 foram acessadas.

Após a interrupção 5: as páginas 1 e 2 foram acessadas.

4 10110000



5 01010000

Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0	1	2	3	4	5
---	---	---	---	---	---

0	1	1	0	0	0
---	---	---	---	---	---

Vamos ver como serão os valores dos contadores, se ocorrerem os seguintes acessos às páginas, em cinco interrupções consecutivas do temporizador:

0 01111000

Após a interrupção 1: as páginas 0, 2, 4, e 5 foram acessadas.

1 10110000

Após a interrupção 2: as páginas 0, 1, e 4 foram acessadas.

2 10001000

Após a interrupção 3: as páginas 0, 1, 3, e 5 foram acessadas.

3 00100000

Após a interrupção 4: as páginas 0 e 4 foram acessadas.

4 01011000

Após a interrupção 5: as páginas 1 e 2 foram acessadas.

5 00101000



Algoritmo de idade



O **algoritmo de idade** propõe as seguintes alterações no algoritmo anterior, para resolver o problema que foi descrito:

- Em cada interrupção, para cada página da tabela:
 - O contador é deslocado um bit para à direita.
 - O bit R é copiado para o bit mais significativo do contador.
- Assim como antes, ao ocorrer uma falha de página, a página com o menor valor para o contador é a escolhida.

Bits referenciada

0 1 2 3 4 5

0	1	1	0	0	0
---	---	---	---	---	---

Se ocorrer uma falha de página após 5 interrupções, o algoritmo escolherá a página com o menor contador, no exemplo, a página 3. Existem duas diferenças entre o algoritmo de idade e o LRU:

0 01111000

- O algoritmo não sabe, entre duas interrupções consecutivas, qual das páginas foi referenciada primeiro. Neste caso, deveremos ver os acessos anteriores à página. Escolhemos a página 3 porque a página 5 foi a única usada na primeira interrupção do temporizador.

1 10110000

- O contador possui um número finito de bits, e com isso, se dois ou mais contadores se tornarem nulos, deveremos escolher uma das páginas com o contador nulo de modo aleatório.

2 10001000

3 00100000

4 01011000

5 00101000



G/G