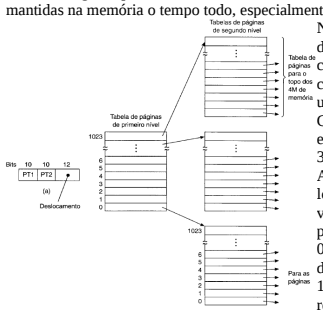
Gerenciamento de Memória

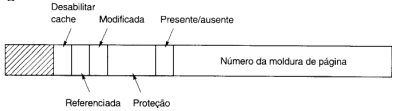
Tabelas de páginas: O objetivo da tabela de páginas é mapear páginas virtuais em molduras de página física. Matematicamente, a tabela de páginas é uma função que usa o número da página virtual como argumento e tem o número da moldura de página física correspondente como resultado. Usando o resultado dessa função, o campo que endereça a página virtual do endereço virtual pode ser substituído pelo campo que endereça a moldura de página física, formando assim um endereço da memória física.
Dois pontos importantes devem ser considerados:

1. A tabela de páginas pode ser extremamente grande.
2. O mapeamento deve ser rápido.

Lembre-se: cada processo necessita de sua própria tabela de paginas.
O segundo ponto advém do fato de o mapeamento virtual-físcio ser feito em todas as referências à memória. Uma instrução típica localiza-se em uma palavra da memória, e muitas vezes cada um de seus operandos também. Consequentemente, é necessário fazer, por instrução, uma, duas ou às vezes mais referências à tabela de páginas virtuais.
O projeto mais simples é ter uma única tabela de páginas que consista em um vetor de registradores rápidos em hardware, com uma entrada para cada página virtual indexada pelo número de página virtual. Quando um processo estiver para ser executado, o Sistema Operacional (SO) carregará esses registradores a partir de uma cópia da tabela de páginas desse processo mantida na memória. Durante a execução desse processo não serão mais necessárias referências à tabela de páginas virtuais da memória. Esse método é vantajoso porque é simples e direto e não requer qualquer referência à memória durante o mapeamento.
No outro extremo, pode-se ter a tabela de páginas totalmente na memória. Todo o hardware necessário resume-se a um único registrador, que aponta para o início da tabela de páginas. Essa prática permite que a modificação do mapeamento de páginas durante o chaveamento de contexto seja feita por meio de uma simples carga de um novo valor no referido registrador, que aponta para o início da tabela de páginas. Essa prática permite que a modificação do mapeamento de páginas durante o chaveamento de contexto seja feita por meio de uma simples carga de um novo valor no referido registrador.
Obviamente, esse método apresenta a desvantagem de requerer uma ou mais referências à memória para ler as entradas da tabela de páginas durante a execução de cada instrução.
Tabelas de página multinível: A fim de minimizar o problema de continuamente armazenar tabelas de páginas muito grande na memória, diversos computadores utilizam tabelas de páginas em múltiplos níveis. O segredo para o método multinível de tabelas de páginas é evitar que todas elas sejam mantidas na memória o tempo todo, especialmente aquelas que não são necessárias.



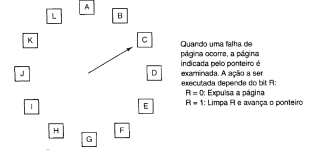
No lado esquerdo da figura vemos a tabela de páginas de nível 1, com 1024 entradas, correspondente ao campo PT1 de 10 bits. Quando um endereço virtual chega à MMU, ela primeiro extra o campo PT1 e o utiliza como índice da tabela de páginas de nível 1. Cada uma dessas 1024 entradas represa 4M, pois o espaço total de endereçamento virtual de 4 Gb (isto é, 32 bits) foi dividido em segmentos de 1024 bytes.
A entrada da tabela de páginas de nível 1, que é localizada por meio do campo PT1 do endereço virtual, aponta para o endereço ou a moldura de página de uma tabela de páginas de nível 2. A entrada 0 da tabela de página de nível 1 aponta para a tabela de página de nível 2 relativa aos dados e a entrada 1023 aponta para a tabela de páginas de nível 2 relativa à pilha. O campo PT2 é então empregado como índice da tabela de páginas de nível 2 selecionada para se localizar o número da moldura de página física correspondente.
A MMU primeiro utiliza PT1 como índice da tabela da página de nível 1 e obtém a entrada 1, a qual corresponde ao endereço de uma tabela de nível 2 que contém os endereços. Os bits *presente/ausente* nas 1021 entradas não usadas da tabela de páginas de nível 1 são marcados com 0, forçando uma falta de página se forem acessadas. Se isso ocorrer, o SO saberá que o processo está tentando referenciar uma parte não permitida da memória e tomará uma decisão apropriada.
Estrutura de uma entrada da tabela de páginas: O esquema exato de uma entrada é altamente dependente da máquina, mas o tipo da informação presente é aproximadamente o mesmo para máquinas. O campo mais importante é o *Número da moldura de páginas.* Afinal de contas, o objetivo do mapeamento de páginas é localizar esse valor.



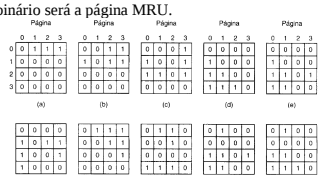
Os bits de *proteção* dizem quais tipos de acesso são permitidos à página.
Os bits *modificada* e *referenciada* controlam o uso da página. Ao se escrever na página, o hardware automaticamente faz o bit *modificada* igual a 1 na entrada correspondente da tabela de páginas. Se a página física dentro dessa moldura foi modificada (isto é, fico "suja"), ela deve ser também atualizada no disco. Do contrário (isto é, se continua "limpa"), ela pode ser simplesmente abandonada, visto que a imagem em disco continua válida. Esse bit é chamado algumas vezes de ***bit sujo***, já que ele reflete o estado da página.
Ao bit *referenciada* é atribuído o valor 1 quando a página física é referenciada, para leitura ou escrita. Esse bit ajuda o SO a escolher uma página a ser substituída quando ocorre uma falta de página. Por fim, o bit *cache desabilitado* permite que o mecanismo de cache seja desabilitado para a página em questão. Essa propriedade é mais importante para as páginas que mapeiam em registradores de dispositivos do que em memória.
A tabela de páginas contém somente as informações necessárias ao hardware para traduzir o endereço vitual em endereços físicos.
Memória associativa ou TLB: Com base na observação de que a maioria dos programas tende a fazer um grande número de referências a um mesmo pequeno conjunto de páginas virtuais.
A solução concebida foi equipar os computadores com um pequeno dispositivo em hardware para mapear os endereços virtuais para endereços físicos sem passar pela tabela de páginas. Esse dispositivo, denominado ***TLB (Translation lookaside Burffer***) ou às vezes ***memória associativa***. Cada entrada contém informações sobre uma página, um bit que é colocado em 1 quando a página é *modificada*, o código de proteção e a moldura de página em que está localizada. Esses campos têm uma correspondência de um para um com os campos da tabela de páginas.
Quando um endereço virtual é apresentado à MMU para a tradução, o hardware primeiro verifica se o número de sua página virtual está presente na TLB comparando-o com todas as entradas da TLB simultaneamente. Se uma correspondência válida é encontrada e o acesso não viola os bits de *proteção*, o número da moldura de página é então obtido diretamente da TLB, sem necessidade de buscá-lo na tabela de páginas. Se o número da página virtual estiver presente na TLB, mas, se a instrução estiver tentando escrever em uma página que permita somente leitura, uma *falta por violação de proteção (protection falut)* é gerada, do mesmo modo que aconteceria no acesso à própria tabela de páginas.
É interessante nota o que acontece quando o número da página virtual não está presente na TLB. A MMU detecta a ausência (*miss)* e então faz uma busca comum na tabela de páginas. A MMU então destitui uma das entradas da TLB e a substitui por essa entrada da tabela de página que acabou de ser buscada. Quando uma entrada é retirada da TLB, apenas o bit *modificada* deve ser copiado de volta na entrada correspondente da tabela de páginas na memória. Os demais valores já estão lá. Mas, quando uma entrada da TLB é carregada a partir da tabela de páginas, todos os campos dessa entrada devem ser trazidos da memória.
Gerenciamento da TLB por software: Quando ocorre uma ausência de página na TLB, em vez de a própria MMU buscar na tabela de página a página virtual requisitada, ela apenas gera uma interrupção (*trap)* e repassa o problema ao SO. O SO deve então encontrar a página virtual na tabela de páginas, destituir uma das entradas da TLB, inserir aí a nova página virtual e reiniciar a instrução interrompida.
O problema em se fazer essa pesquisa por software é que as áginas que contêm a tabela de páginas podem não estar mapeadas na TLB, ocasionando ausências adicionas na TLB durante esse tratamento.
Tabelas de páginas invertidas: É necessário uma solução diferente para tratar espaços de endereçamento virtual paginado de 64 bits. Uma possível solução é a tabela de páginas invertidas: nela existe apenas entrada por moldura de página na memória real, em vez de uma entrada por página do espaço de endereçamento virtual.
Algoritmos de Substituição de páginas: Quando uma falta de página ocorre, o SO precisa escolher uma página a ser removida da memória a fim de liberar espaço para um nova página a ser trazida para a memória. Se a página a ser removida tiver sido modificada enquanto esteve na memória, ela deverá ser reescrita no disco com o propósito de atualiza a cópia virtual lá existente. Se, contudo, a página *não* tiver sido modificada, a cópia em disco já estará atualizada e, assim, não será necessário reescrevê-la. A página a ser trazida para a memória simplesmente sobrepões a página que está sendo destituída.
O algoritmo de substituição de página *ótimo:* No momento em que ocorre uma falta de página, existe um determinado conjunto de páginas na memória. Uma delas será referenciada na próxima instrução, ou seja, trata-se da mesma página que contém a instrução que gerou a falta de página. Cada página pode ser rotulada com o número de instruções que serão executadas antes de aquela página ser referenciada pela primeira vez. Na ocorrência de uma falta de página, o SO não tem como saber quando cada uma das páginas será novamente referenciada.
O algoritmo de substituição de página não usada recentemente (NUR): A maioria dos computadores com memória virtual tem 2 bits de status – o bit *referenciada (R)* e o bit *modificada (M)* –, associados a cada página virtual, que permitem que o SO saiba quais páginas físicas estão sendo usadas e quais não estão. O bit R é colocado em 1 sempre que a página é *referenciada*. O bit M é colocado em 1 sempre que se escreve na página (isto é, a página é *modificada*). É importante perceber que esses bits devem ser atualizados em todas as referências à memória, de modo que é essencial que essa atualização se dê por hardware.
Se o hardware não possui esses bits, estes podem ser simulados da seguinte maneira: um processo, ao ser inciado, tem todas as suas entradas da tabela de páginas marcadas como não presente na memória. Tão logo uma de suas páginas virtuais seja referenciada, ocorre uma falta de páginas. O SO então coloca o bit R em 1 (em suas tabelas internas), altera a entrada da tabela de páginas a fim de apontar para a páginas física correta, com modo SOMENTE LEITURA, e reinicia a instrução. Se a página for subsequentemente escrita, outra falta da página ocorrerá, permitindo que o SO coloque o bit M em 1 e altere o modo da página para LEITURA/ESCRITA.
Quando um processo é inciado, os dois bits citados, para todas as suas páginas, são colocados em 0 pelo SO. Periodicamente, o bit R é limpo de modo que diferencie as páginas que não forma referenciadas recentemente daquelas que foram. Quando acontece uma falta de página, o SO inspeciona todas as páginas e as separa em quatro categorias, com base nos valores atuais dos bit R e M:

- Classe 0: não referenciada, não modificada.
- Classe 1: não referenciada, modificada.
- Classe 2: referenciada, não modificada.
- Classe 3: referenciada, modificada.

Embora as páginas da classe 1 pareçam, à primeira vista, impossíveis de ocorrer, elas surgem quando uma página da classe 3 tem seu bit R limpo por uma interrupção do relógio.
O ***algoritmo NUR*** (não usado recentemente) remove aleatoriamente uma página da classe de ordem mais baixa que não esteja vazia. Está implícito nesse algoritmo que é melhor remover uma página modificada mas não referenciada pelo menos uma interrupção do relógio (em geral 20ms) do que uma página não modificada que está sendo intensamente referenciada.
O algoritmo de substituição de página primeira a entrar, primeira a sair (FIFO): O SO mantém uma lista de todas as páginas atuais na memória, com a página mais antiga na cabeça e a página que chegou mais recentemente sutada no final da lista.
O algoritmo de substituição de página segunda chance (SC): Este algoritmo procura uma página antiga que não tenha sido referenciada no intervalo de relógio anterior. Se todas as páginas foram referenciadas, o SC degenera-se para o FIFO puro.
O algoritmo de substituição de página relógio: Uma estratégia melhor é manter todas as páinas em uma lista circular em forma de relógio. Quando ocorrer uma falta de página, a página mais antiga, apontada pelo ponteiro, será inspecionada.



O algoritmo de substituição da página menos recentemente usada (MRU): páginas referenciadas intensamente nas últimas instruções provavelmente serão de novo referenciadas de maneira intensa nas próximas instruções. Quando ocorre uma falta de página, descartamos a página menos recentemente usada no fim da lista. A desvantagem é que essa lista encadeada deve ser atualizada a cada referencia à memória. Contudo existem outras maneira de implementar o MRU, vamos considerar o modo mais simples primeiro: ele requer que o hardware seja equipado com um contador de 64bits, que é incrementado automaticamente após a execução de cada instrução. Além disso, cada entrada da tabela de páginas deve ser um campo extra para armazenar o valor do contador. Após cada referência à memória, o valor atual do contador é armazenado nesse campo adiciona da entrada da tabela de paginas correspondente à página que acabou de ser referenciada. A página correspondente a esse menor valor será a "menos recentemente usada".
Uma segunda maneira consistem construir uma matriz de bits n X n. Em um instante qualquer, a linha que possuir o menor valor binário será a página MRU.



Simulação do MRU em software: É preciso encontrar uma solução implementável em software. Uma possibilidade é empregar o algoritmo de substituição de página não usada frequentemente (NUF). A implementação desse algoritmo requer contadores em software, cada um deles associado a uma página, inicialmente zerada. A cada interrupção de relógio, o SO percorre todas as páginas na memória. Para cada página, o bit R, que pode estar em 0 ou 1, é adicionado ao contador correspondente.
Primeiramente, os contadores são deslocados um bit à direita. Em seguida, o bit R de cada página é adicionado ao bit mais à esquerda do contador correspondente, em vez de ao bit mais à direita.