Sistemas Operacionais

Prof. Robson de Souza

Aulas 27 e 28

Conteúdo: Gerência de memória: Memória virtual e paginação

Em um Sistema Operacional, a CPU deve buscar e executar instruções que estão na memória principal, porém, essa memória tende a ser muito pequena para armazenar a grande quantidade de dados que esses programas possuem. Para resolver isso, foi necessária a implementação de uma hierarquia de memória e com isso o S.O utiliza um sistema chamado de **memória virtual**.

Memória virtual é uma técnica onde as memórias principal e secundária são combinadas dando ao usuário a ilusão de existir uma memória maior que a capacidade real da memória principal. Desta forma, quando a memória principal está cheia e não há mais espaço para novos programas ou dados, o sistema utiliza a memória secundária. Tudo é feito de forma automática pela Unidade de Gerência de Memória (ou Memory Management Unit - MMU) presente nos processadores. Assim, todo dado que é acessado é antes buscado pela MMU na memória principal. Se ele não estiver lá, ela vai buscar na memória secundária, faz uma cópia na memória principal e libera o acesso ao dado.

Esse modelo permite que o programador enderece grandes quantidades de dados, deixando para o sistema a responsabilidade de transferir o dado da memória secundária para a principal. A organização do fluxo entre a memória principal e a secundária deve ser feita, o que consiste em transformar o endereço usado pelo programador na localização física de memória correspondente.

Espaço de endereçamento → Endereços usados pelo programador.
Espaço de memória → Localizações de memória no computador.

O espaço de endereçamento N e o espaço de memória M podem ser vistos como um mapeamento de endereços do tipo: $f:N \rightarrow M$.
O mapeamento permite ao programador usar um espaço de endereçamento maior que o espaço de memória principal disponível.

A memória virtual possui o **sistema de paginação**, onde o espaço de endereçamento é dividido em páginas de tamanho igual, em geral múltiplos de 512 KB. A memória principal é dividida em molduras de páginas de tamanho igual. As molduras de páginas contém algumas páginas ativas, enquanto o restante das páginas estão residentes em memória secundária (páginas inativas).

O mecanismo de paginação possui duas funções:

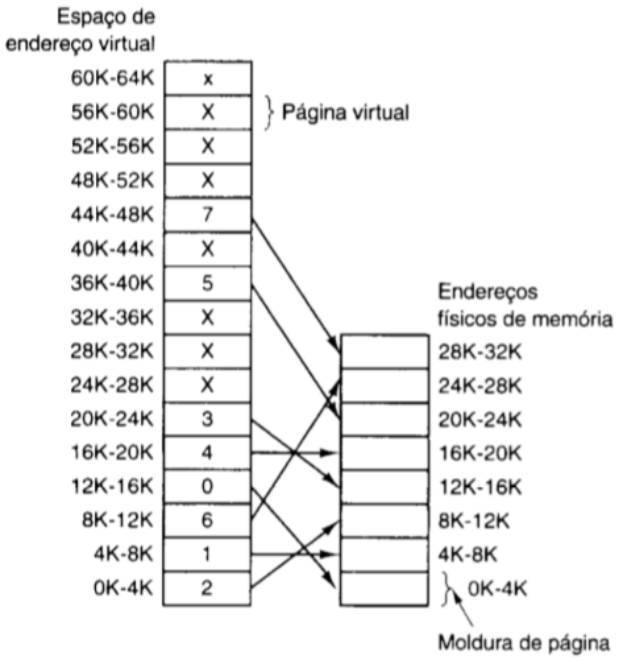
Mapear os endereços → Determinar qual página um programa está endereçando e encontrar a moldura se existir, que contenha a página.

Transferência de páginas → Transferir páginas da memória secundária para a memória principal e transferi-las de volta para a memória secundária quando não estão mais sendo utilizadas.

As páginas e as molduras de páginas possuem exatamente o mesmo tamanho, com isso, é feito um mapeamento que indica a qual moldura de página cada página pertence.

Obviamente que um programa pode não utilizar exatamente todos os bytes presentes em uma página, com isso, deve ser fornecido um valor que indica qual byte dentro daquela página o programa quer acessar.

A figura abaixo mostra um exemplo de mapeamento entre os endereços virtuais de memória e os endereços físicos:



Nesse exemplo, se o programador tentar acessar o endereço 0 (endereço virtual), a unidade de gerenciamento de memória mapeará esse endereço para a moldura de página 2, onde estão os endereços físicos de 8K a 12K.

Porém, se o programador tentar acessar o endereço 5120, o gerenciador de memória mapeará esse endereço virtual para a página 1, que abrange 4K a 8K, que por sinal possui os mesmos endereços físicos. A questão aqui é que o endereço 5120 corresponde a 1024 bytes além de 4K (4096). Isso mostra que além do número da página, é preciso indicar qual byte dentro dessa mesma página deve ser acessado.

Com isso, o endereçamento de uma página é feito de modo que uma parte dos bits é interpretada como um número de página e a outra parte como o número do byte dentro da página (offset).

O mapeamento dos endereços é feito através de uma **tabela de páginas**, onde a p-ésima entrada da tabela contém a localização p' da moldura de página contendo a página número p desde que esteja na memória principal. O mapeamento de endereços é:

$$f(e) = f(p,b) = p' + b$$

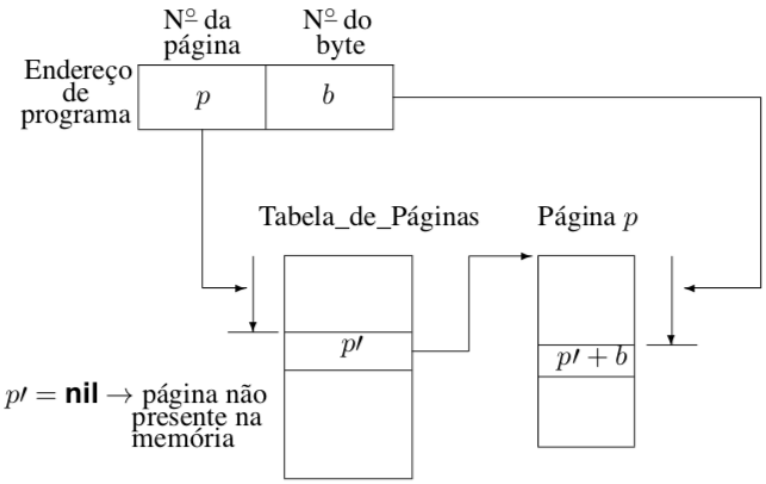
onde:

- e → Endereço do programa
- p → número da página
- b → número do byte

O número de página é usado como um índice da tabela de páginas, que fornece o número da moldura de página correspondente a essa página virtual. O propósito da tabela de páginas é mapear páginas virtuais em molduras de página. Matematicamente falando, a tabela de páginas é uma função, com o número de página virtual como argumento, e o número da moldura física como resultado. Utilizando o resultado dessa função, o campo de página virtual em um endereço virtual pode ser substituído por um campo de moldura de página, formando assim, um endereço físico de memória.

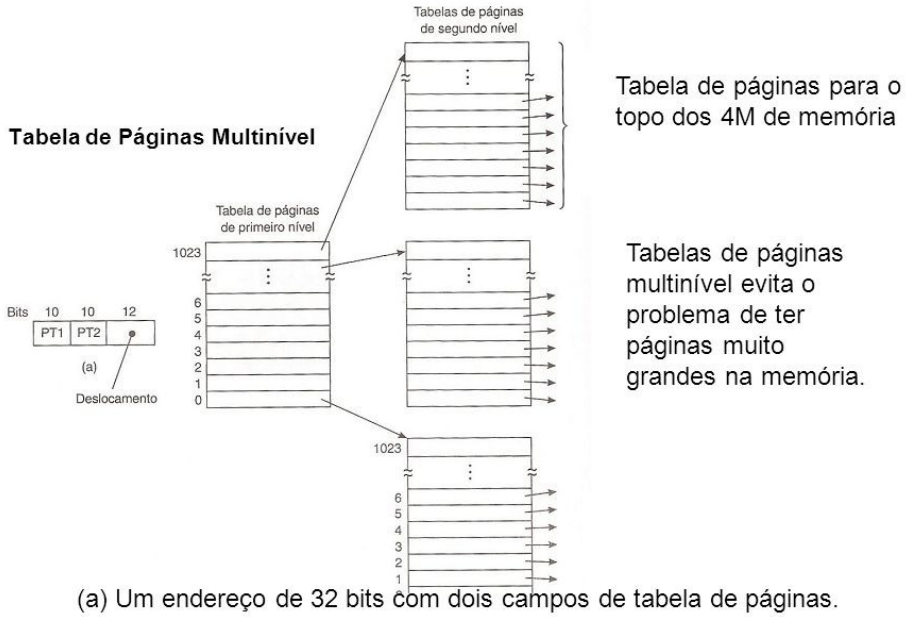
É importante ressaltar duas coisas:

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



Com relação às tabelas de páginas serem muito grandes, pode ocorrer desperdício. Por exemplo, imagine um processo que necessite apenas de 32 bits de dados em uma página (4 bytes) que possui 4KB, nesse caso, a memória fica com 4092 bytes em uso completamente ociosos.

Uma solução para isso é a chamada **paginação multinível**, que consiste em dividir uma tabela de páginas em mais níveis menores, com isso, o campo de endereços pode ser dividido em partes que representam os níveis de cada tabela. Nesse caso, as entradas das tabelas de primeiro nível representam tabelas de segundo nível e assim por diante.



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

Para fazer o mapeamento de forma rápida, pode-se utilizar um dispositivo chamado **Translation Lookaside Buffer (TLB)** ou **memória associativa**. Esse dispositivo normalmente está dentro da MMU e consiste em um pequeno número de entradas que contém as informações sobre uma página, como por exemplo, o número de página virtual, um bit para indicar quando a página é modificada, código de proteção, moldura da página física, entre outros.

*Políticas de busca e alocação de páginas

O mecanismo de memória virtual permite que a execução de um programa sem que seu código esteja completamente residente na memória principal. Cada processo vai precisar de determina(s) páginas para executar, com isso, o S.O deve gerenciar as buscas e alocação de páginas para cada processo.

Paginação por demanda → Páginas são alocadas na M.P. somente quando referenciadas.

Paginação antecipada → São alocadas a página referenciada e mais algumas páginas que poderão ser necessárias para a execução do programa.

Alocação fixa → Cada processo tem um número máximo de frames que pode ser utilizada na execução do programa. O limite de páginas é definido no momento da criação do processo e é uma informação do contexto de software.

Alocação variável → Depende da disponibilidade da M.P. Apesar de ser flexível este mecanismo obriga o s.o. a monitorar constantemente os processos.

Referências bibliográficas:

TANENBAUM, Andrew. 2ª ed. **Sistemas Operacionais Modernos,** Editora Pearson, 2003.

SILBERSCHATZ, Abraham. **Sistemas Operacionais com JAVA,** 6ª ed. Editora Campus

MACHADO, Francis B. **Arquitetura de Sistemas Operacionais,** 4ª ed, LTC, 2007.