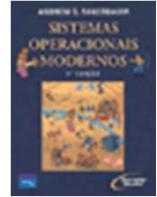


Capítulo 4

Gerenciamento de Memória

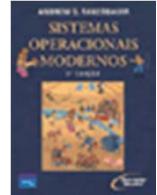
- 4.1 Gerenciamento básico de memória
- 4.2 Troca de processos
- 4.3 Memória virtual
- 4.4 Algoritmos de substituição de páginas
- 4.5 Modelagem de algoritmos de substituição de páginas
- 4.6 Questões de projeto para sistemas de paginação
- 4.7 Questões de implementação
- 4.8 Segmentação

Gerenciamento de Memória



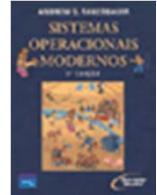
- Idealmente, o que todo programador deseja é dispor de uma memória que seja
 - grande
 - rápida
 - não volátil
- Hierarquia de memórias
 - pequena quantidade de memória rápida, de alto custo - cache
 - quantidade considerável de memória principal de velocidade média, custo médio
 - gigabytes de armazenamento em disco de velocidade e custo baixos
- O gerenciador de memória trata a hierarquia de memórias

Gerenciamento de Memória



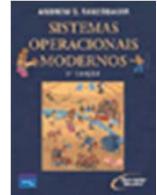
A memória é um dos recursos mais importantes, escassos e caro de um computador, razão pela qual precisa ser gerenciada de uma maneira eficiente. Mesmo que, os computadores atuais possuam mais memória que os existentes no início dos anos 60, o tamanho dos programas está crescendo a uma taxa maior que a taxa de crescimento das memórias. Na memória principal residem todos os programas e dados que serão executados ou referenciados pelo processador, não confundir memória principal com memória secundária. A memória secundária, normalmente disco ou fita, é um meio permanente, com mais capacidade e de baixo custo, onde são armazenados programas e dados. Toda vez que desejarmos executar um programa residente na memória secundária deveremos de alguma forma, carregá-lo para a memória principal.

Gerenciamento de Memória



A parte do sistema operacional que gerencia a memória é denominada gerente de memória, sua função é o de controlar quais partes da memória estão sendo usadas e quais não estão, de forma a alojar memória a processos quando estes precisarem, liberar a memória que estava sendo ocupada por um processo que terminou e tratar problema entre a memória principal e o disco, quando a memória não for o grande suficiente para guardar todos os processos como,.. do swapping e, da paginação e segmentação (usados na implementação de memória virtual).

Gerenciamento de Memória



Nos sistemas monoprogramável a gerência de memória é simples, já nos sistemas multiprogramável ela é muito complexa e se torna crítica. Devido à necessidade de se manter o maior número de usuários possível utilizando a memória eficientemente, tornando sua gerência muito mais difícil.

Os sistemas de gerência de memória podem ser divididos em duas grandes categorias: aqueles que movem os processos entre a memória principal e o disco (swapping, paginação e segmentação), e aqueles que não movimentam os processos entre tais dispositivos de armazenamento.

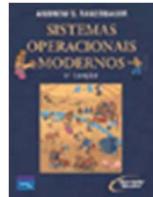
Gerenciamento de Memória Sem Swapping ou Paginação



Gerência de Memória sem Swapping ou Paginação

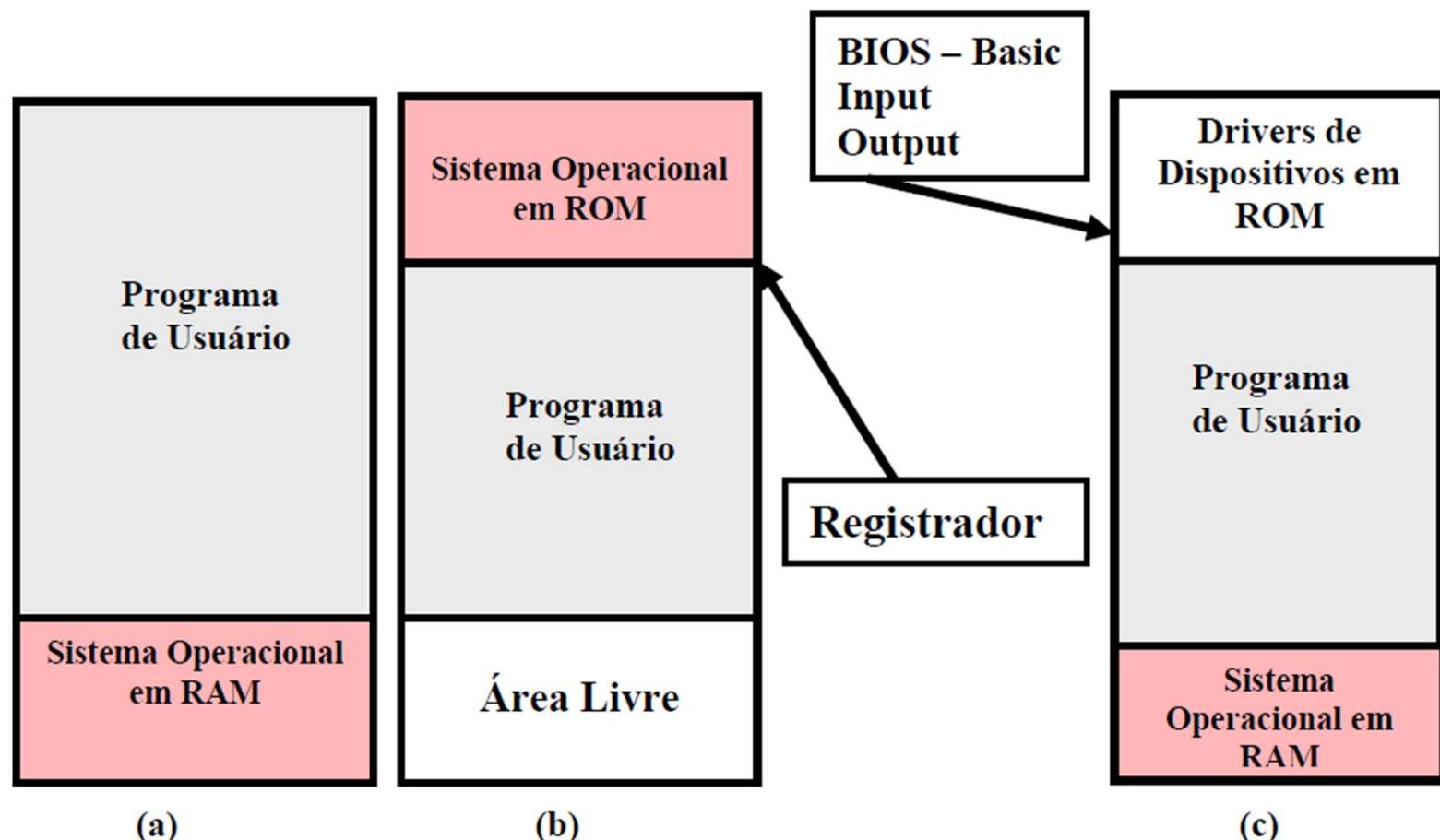
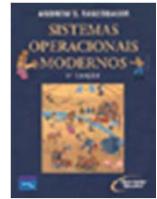
O esquema mais simples de alocação de memória a processos, usado até 1960, é aquele no qual só existe um único processo na memória em cada instante, sendo permitido que tal processo use toda a memória disponível, a monoprogramação. O usuário carrega a memória principal com um processo vindo do disco ou d fita magnética, e este assume o controle de todos os recursos da máquina. Atualmente, nos computadores pessoais, os processos precisam de um driver para cada dispositivo de E/S e o código destes drivers ter de estar presente na memória junto com o código do processo.

Gerenciamento de Memória Sem Swapping ou Paginação Alocação Contígua Simples



Nesse tipo de gerência de memória, a memória principal é dividida em duas partes: uma para o sistema operacional e outra para o programa do usuário. Dessa forma, o programador deve desenvolver suas aplicações, preocupado, apenas, em não ultrapassar o espaço de memória disponível, ou seja, o tamanho total da memória principal menos o que está sendo ocupado pelo SO.e pelo processo usuário, foi implementada nos primeiros sistemas operacionais, porém ainda está presente em alguns sistemas monoprogramáveis. Observa-se , (a) O sistema operacional pode estar no início da memória RAM, (b) pode estar na parte mais alta do sistema de endereçamento da memória em ROM e (c) os drivers de dispositivos em ROM, ocupando uma determinada quantidade de memória alta, conhecido com o nome BIOS (Basic Input Output System) e o restante do sistema operacional em RAM.

Gerenciamento de Memória Sem Swapping ou Paginação Alocação Contígua Simples



Gerenciamento de Memória Sem Swapping ou Paginação Alocação Contígua Simples



Neste tipo de gerência de memória, o usuário tem controle sobre toda a memória principal, podendo acessar qualquer posição de memória, inclusive para alterar e destruir o SO. Para protegê-lo desses ataques, que podem ser conscientes ou não, alguns sistemas implementam proteção através de um registrador, que delimita as áreas do SO e do usuário. Dessa forma, sempre que um programa de usuário faz referência a um endereço na memória, o sistema verifica se o endereço está nos seus limites. Caso não esteja, o programa do usuário é cancelado e uma mensagem de erro é enviada (violação de acesso- access violation).

Apesar de sua fácil implementação e código reduzido, a alocação simples não permite a utilização eficiente do processador e da memória, pois apenas um usuário pode dispor desses recursos. Em relação à memória, caso o programa do usuário não a preencha totalmente, existirá um espaço de memória sem utilização.

Gerenciamento de Memória Sem Swapping ou Paginação Técnica de Overlay



A técnica de Overlay (sobreposição), consiste em dividir o programa em partes (módulos) devido a que os programas dos usuários estavam limitados ao tamanho da memória principal disponível, de forma que pudessem executar independentemente uma da outra utilizando uma mesma área de memória.

Dado o seguinte exemplo, um programa que tenha três módulos: um módulo que corresponde ao programa principal, um módulo de cadastramento e um módulo de impressão. Sendo que os módulos de cadastramento e impressão são independentes, quando um módulo estiver na memória, o outro não precisa necessariamente estar. O módulo do programa principal é comum aos dois módulos, devendo permanecer na memória durante todo o tempo.

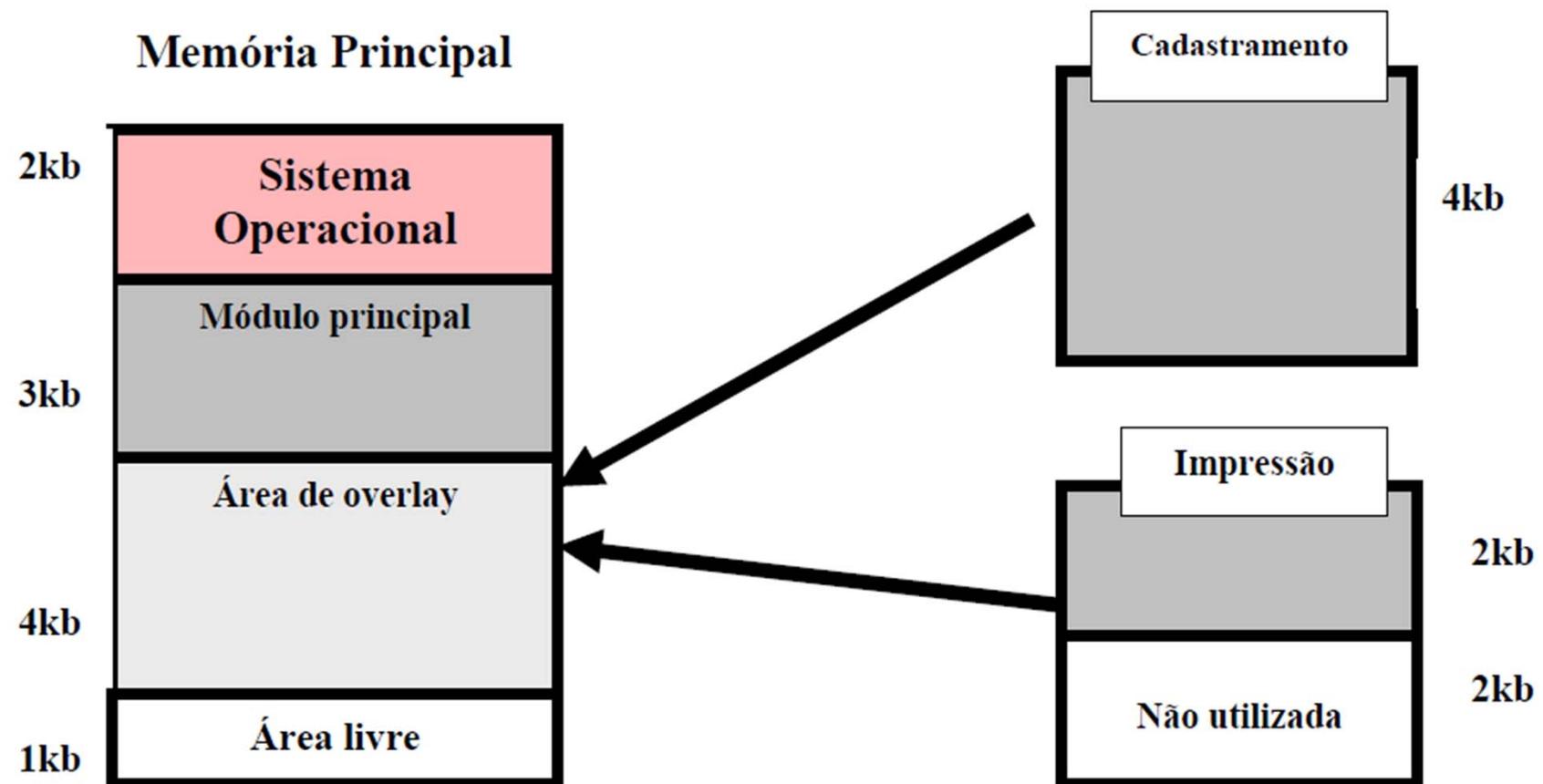
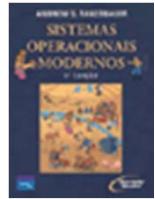
Gerenciamento de Memória Sem Swapping ou Paginação Técnica de Overlay



A memória é insuficiente para todo o programa a técnica de overlay utiliza uma área de memória comum, onde os módulos de cadastramento e impressão poderão compartilhar a mesma área de overlay. Cada vez que um dos dois módulos for referenciado pelo módulo principal, o módulo será carregado da memória secundária para a área de overlay. No caso de uma referência a um módulo que já esteja na área de overlay, a carga não é realizada, caso contrário, o novo módulo irá sobrepor-se ao que já estiver na memória.

A definição das áreas de overlay é função do programador, através de comando especiais da linguagem utilizada. O tamanho de uma área de overlay será estabelecido a partir do tamanho do maior módulo.

Gerenciamento de Memória Sem Swapping ou Paginação Técnica de Overlay



Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Estática



ABSOLUTA

A alocação particionada estática foi utilizada nos primeiros sistemas multiprogramáveis, a memória era dividida em pedaços de tamanho fixo, chamados partições. O tamanho das partições era estabelecido na fase de inicialização do sistema, em função do tamanho dos programas a serem executados. Para a alteração de tamanho da partição, sempre que fosse necessária, o sistema deveria ser desativado e reinicializado com uma nova configuração.

Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Estática



ABSOLUTA

Tabela de partições

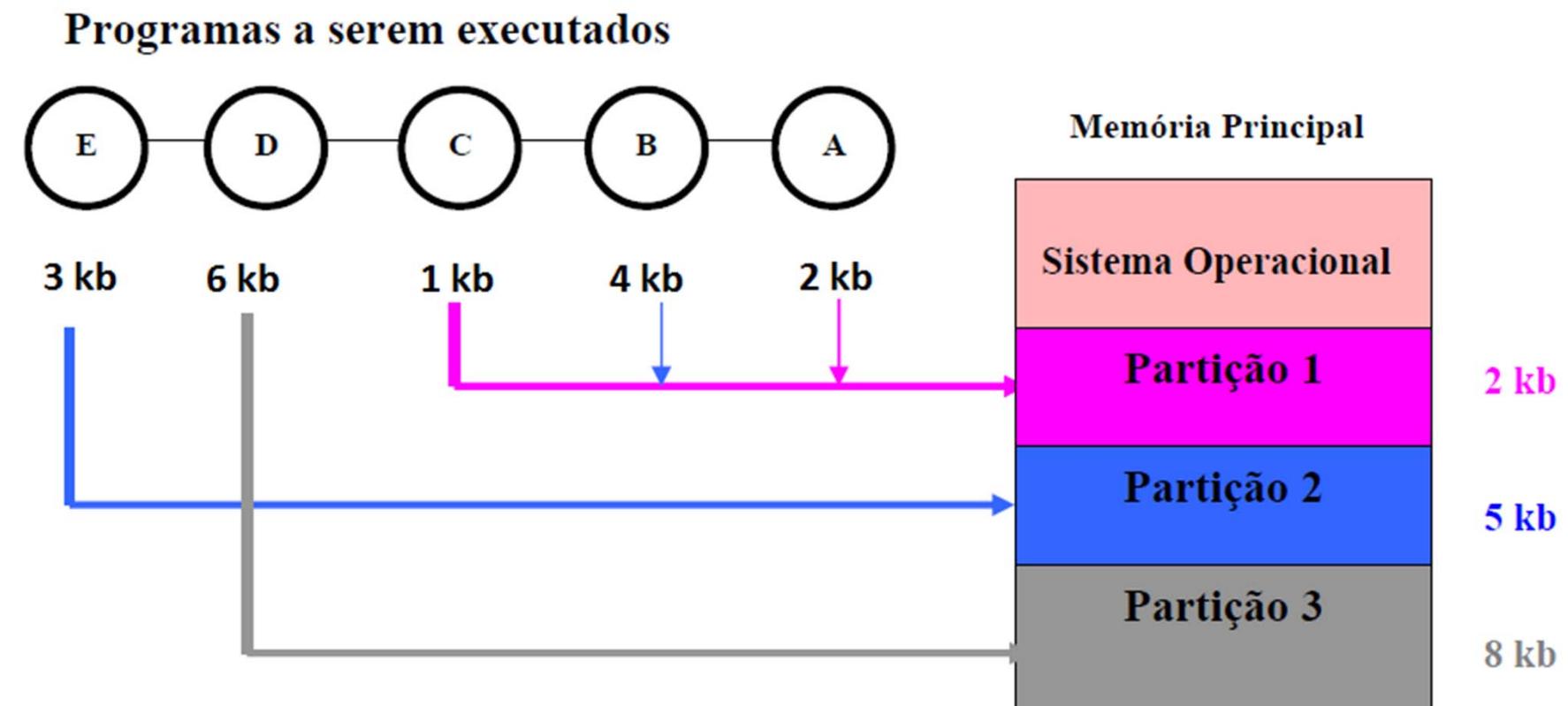
Partição	Tamanho
1	2 kb
2	5 kb
3	8 kb

Memória Principal



Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Estática

ABSOLUTA



Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Estática



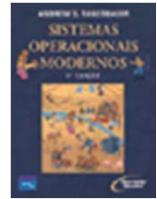
RELOCÁVEL

Os compiladores, linkers e loaders, nesta fase, geram código relocável e os programas puderam ser carregados em qualquer partição. A partir desta implementação, foi criado um novo tipo de organização, denominado alocação particionada estática relocável. se os programas A e B terminassem, o programa E poderia ser executado em qualquer uma das duas partições .

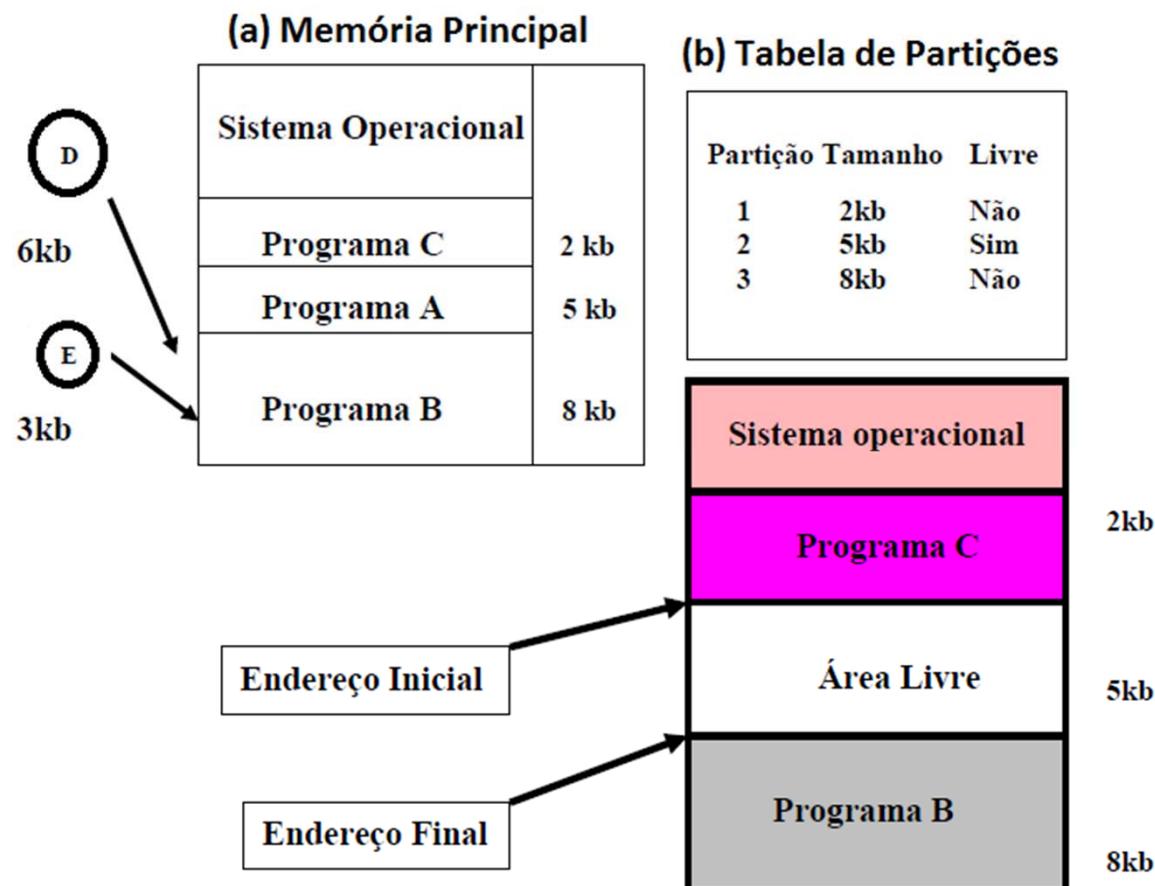
Para manter o controle sobre quais partições estavam alocadas ou não, os sistemas possuíam uma tabela, delimitando cada partição, seu tamanho e se estava em uso não. Sempre que um programa era carregado para a memória, o sistema percorria a tabela na tentativa de localizar uma partição livre, onde o programa pudesse ser alocado Tabela de partições

Nesse esquema de alocação de memória baseia-se em dois registradores, que indicam os limites inferior e superior da partição onde o programa está sendo executado. Caso o programa tente acessar um posição de memória fora dos limites definidos pelos registradores, ele é interrompido e uma mensagem de erro é enviada

Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Estática



RELOCÁVEL

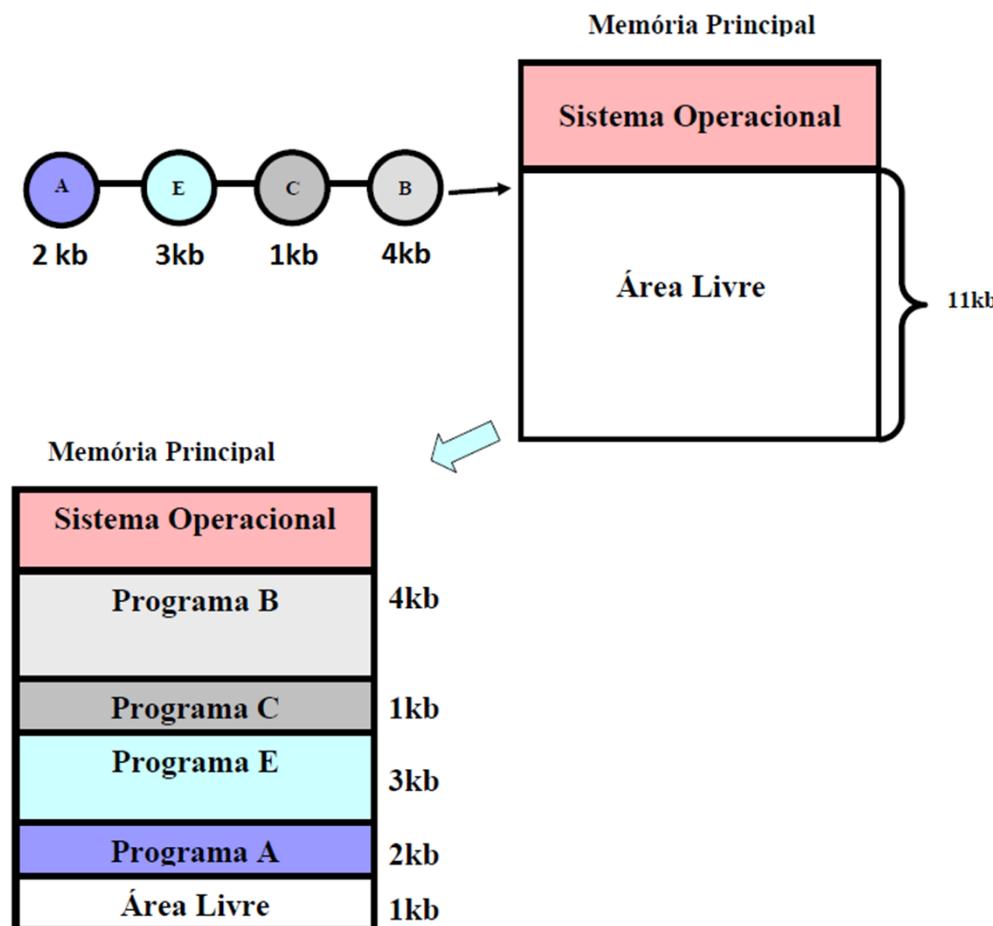


Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Dinâmica

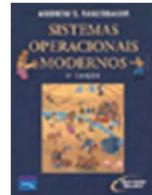


Na alocação particionada estática foi apresentado o problema da fragmentação (“buracos” criados nas partições quando os programas não ocupam a partição toda), portanto foi necessário um outro tipo de alocação como solução a alocação particionada dinâmica, onde foi eliminado o conceito de partições de tamanho fixo. Nesse esquema, cada programa utilizaria o espaço que necessitasse, passando esse pedaço a ser sua partição . Esta alocação diminui o problema da fragmentação para aumentar o uso compartilhado da memória (não confundir com memória compartilhada).

Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Dinâmica

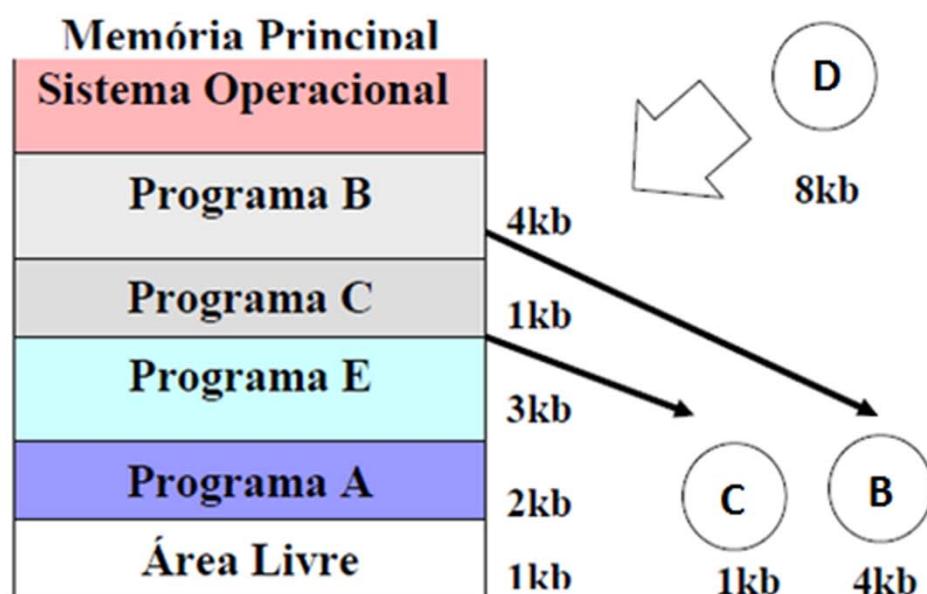


Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Dinâmica



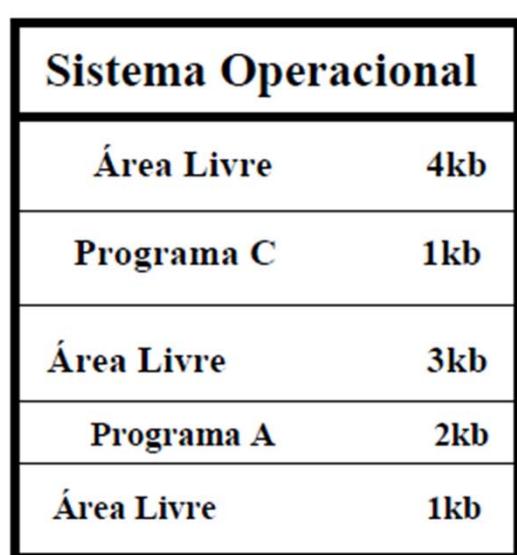
Tanto nos sistemas de alocação absoluta quanto nos de alocação relocável, os programas normalmente, não preenchiam totalmente as partições, onde eram carregados. Além disso, se um programa for maior que qualquer partição livre, ele ficará aguardando um que o acomode, mesmo que existam duas ou mais partições adjacentes que, somadas totalizem o tamanho do programa. Esse tipo de problema, onde pedaços de memória ficam impedidos de serem utilizados por outros programas, é chamado fragmentação. Fragmentação começará a ocorrer, realmente, quando os programas forem terminando e deixando espaços cada vez menores na memória, não permitindo o ingresso de novos programas.

No caso da figura com o término de B e C, e mesmo existindo 8kb livres de memória, o programa D de 6kb não poderá ser carregado

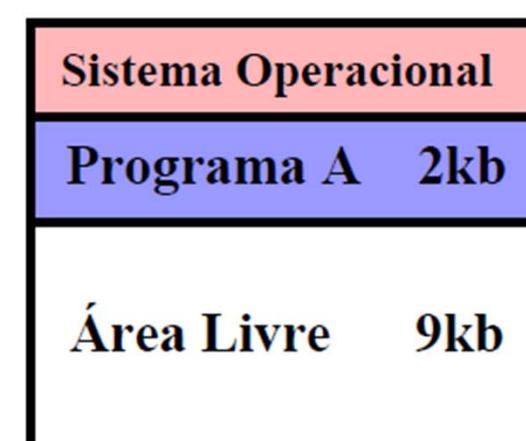


Gerenciamento de Memória Sem Swapping ou Paginação Alocação Particionada Dinâmica

Depois de já ter sido detectada a fragmentação da memória existem duas soluções para o problema, chamado desfragmentação: quando o programa c termina, (a) o sistema pode reunir apenas os espaços adjacentes, produzindo espaço de memória tamanho maior, (b) o sistema pode realocar todas as partições ocupadas, eliminando todos os espaços entre elas e criando uma única área livre contígua (compactação). A complexidade do algoritmo de desfragmentação e o consumo de recursos do sistema, como processador e área em disco, podem torná-lo inviável.

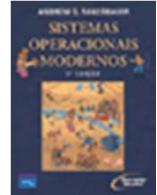


Solução (a)



Solução (b)

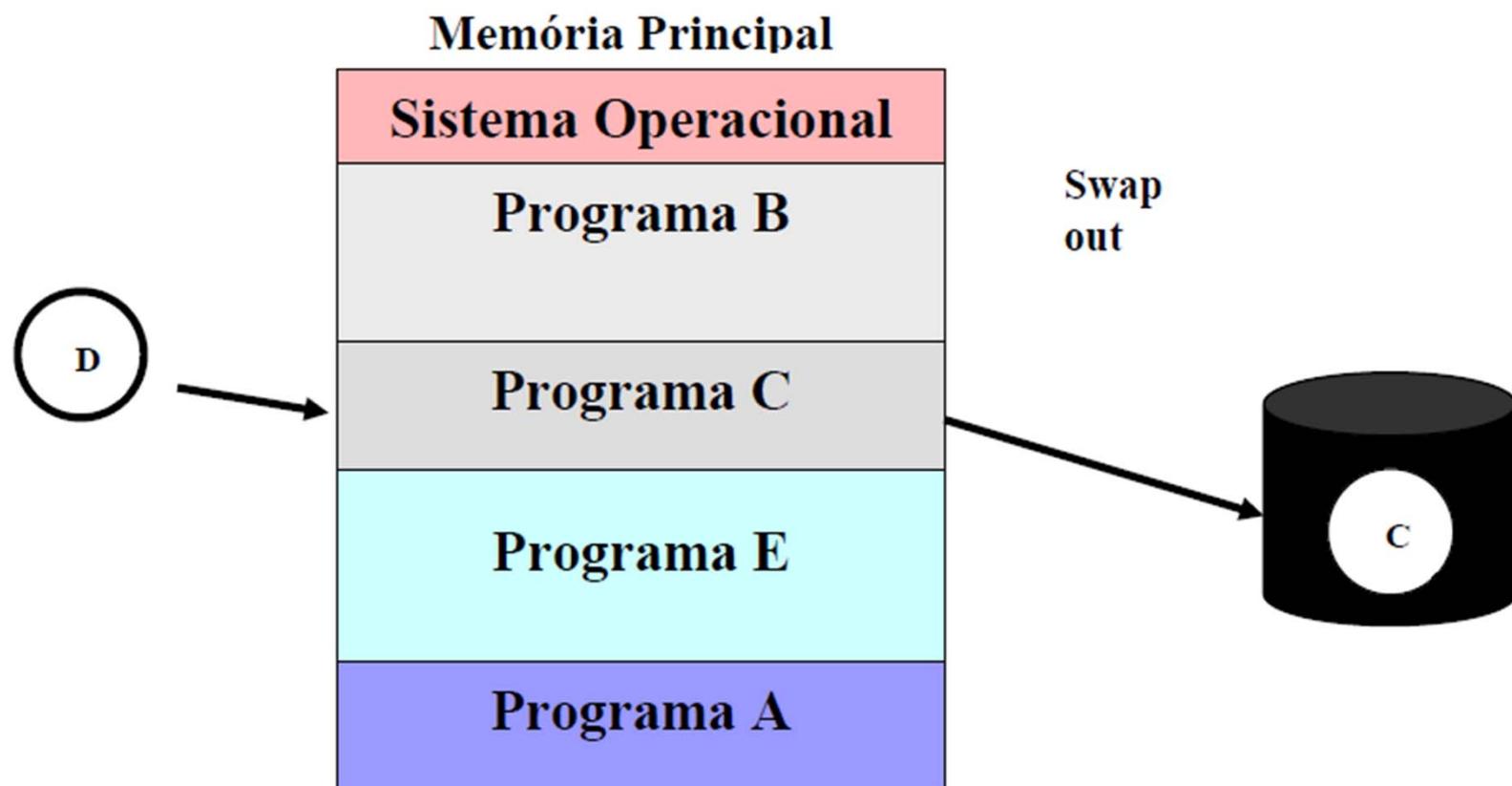
Gerenciamento de Memória Com Swapping



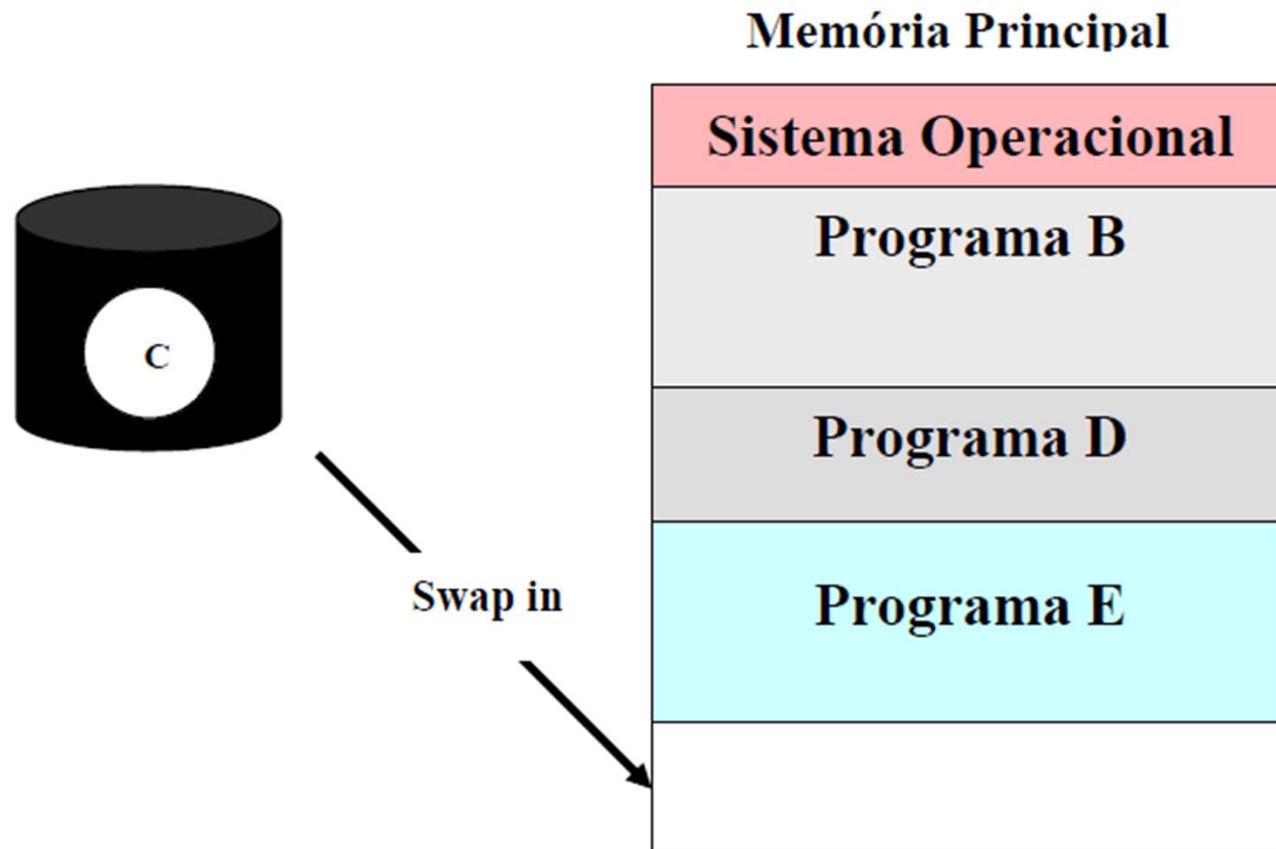
Mesmo tendo um aumento na eficiência da multiprogramação e na gerência de memória, muitas vezes um programa não podia ser executado por falta de uma partição livre disponível. Em todos os esquemas apresentados anteriormente, um programa permanecia na memória principal até o final da sua execução, inclusive nos momentos em que esperava um evento, com uma operação de leitura ou gravação em periféricos.

O swapping é uma técnica aplicada à gerência de memória, para programas que esperam por memória livre para serem processados. Tenta resolver o problema da insuficiência de memória para todos os usuários. Nesta situação, o sistema escolhe um programa residente, que é levado da memória para o disco (swap out), retornando posteriormente para a memória principal (swap in), como se nada tivesse ocorrido.

Gerenciamento de Memória Com Swapping



Gerenciamento de Memória Com Swapping



Gerenciamento de Memória Com Swapping

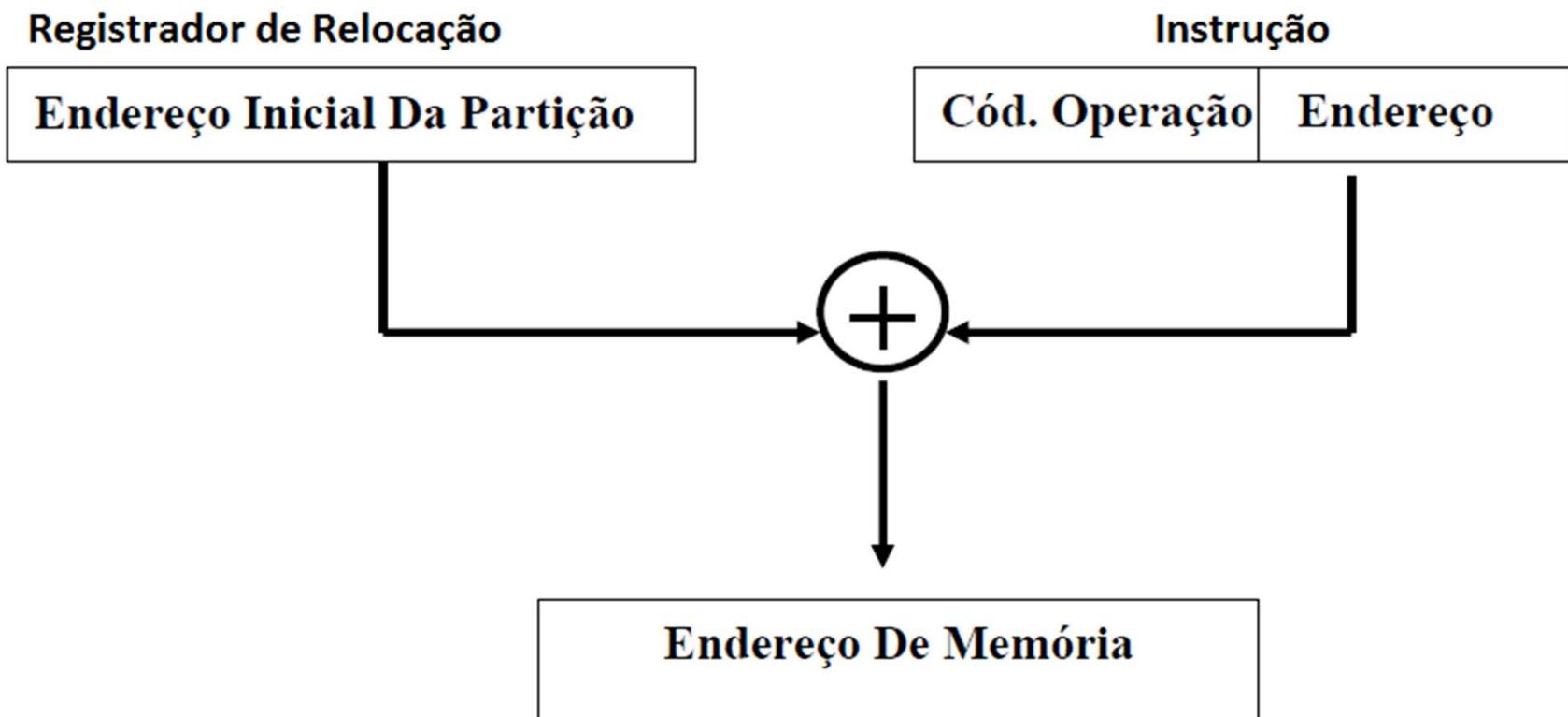


Um dos problemas gerados pelo swapping é a [relocação dos programas](#). O [carregador](#) (loader) relocável permite que um programa seja colocado em qualquer posição de memória, porém a realocação é realizada no momento do [carregamento](#). No caso de um programa que saia e volte muitas vezes para a memória, é necessário que a relocação seja realizada pelo loader a cada carregamento. Essa situação torna o mecanismo [ineficiente](#) em função do tempo gasto para o carregamento. Outra alternativa, também pouco eficiente, é esperar que a região usada pelo programa na ocasião do seu carregamento esteja novamente disponível. A melhor solução para este problema é uma implementação no hardware dos computadores, para permitir que a relocação seja realizada durante a [execução](#) do programa. Esse tipo de mecanismo é denominado [relocação dinâmica](#).

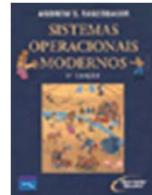
A relocação dinâmica é realizada através de um registrador especial denominado [registrador de relocação](#). No momento em que o programa é carregado na memória, o registrador recebe o endereço inicial da região de memória que o programa irá ocupar. Toda vez que ocorrer uma referência a algum endereço, o endereço contido na instrução será somado ao conteúdo do registrador, gerando assim, o endereço físico. Dessa forma, um programa pode ser carregado em qualquer região da memória. A relocação dinâmica é essencial para a implementação de um sistema multiprogramável.

O conceito de swapping permitiu um maior compartilhamento da memória e um aproveitamento maior da CPU. A técnica se mostrou eficiente para sistemas onde existiam poucos usuários competindo por memória e em ambientes que trabalhavam com aplicações pequenas. Seu maior problema é o elevado custo das operações de entrada/saída (swap in/out).

Gerenciamento de Memória Com Swapping



Gerenciamento de Memória Com Swapping Memória Virtual



Memória virtual é uma técnica onde a memória principal e a secundária são combinadas, dando ao usuário a ilusão de existir uma memória muito maior que a memória principal. O conceito de memória virtual está baseado em desvincular o endereçamento feito pelo programa dos endereços físicos da memória principal. Assim, os programas e suas estruturas de dados deixam de estar limitados ao tamanho da memória física disponível. A memória virtual também veio minimizar o problema da fragmentação.

O conceito de memória virtual se assemelha a idéia de um vetor, existente nas linguagens de alto nível. Quando se faz referência a um componente do vetor, não se sabe em que posição de memória aquele dado está armazenado, é o compilador que se encarrega de gerar instruções que implementam o mecanismo de encontrá-lo.

Gerenciamento de Memória Com Swapping Memória Virtual

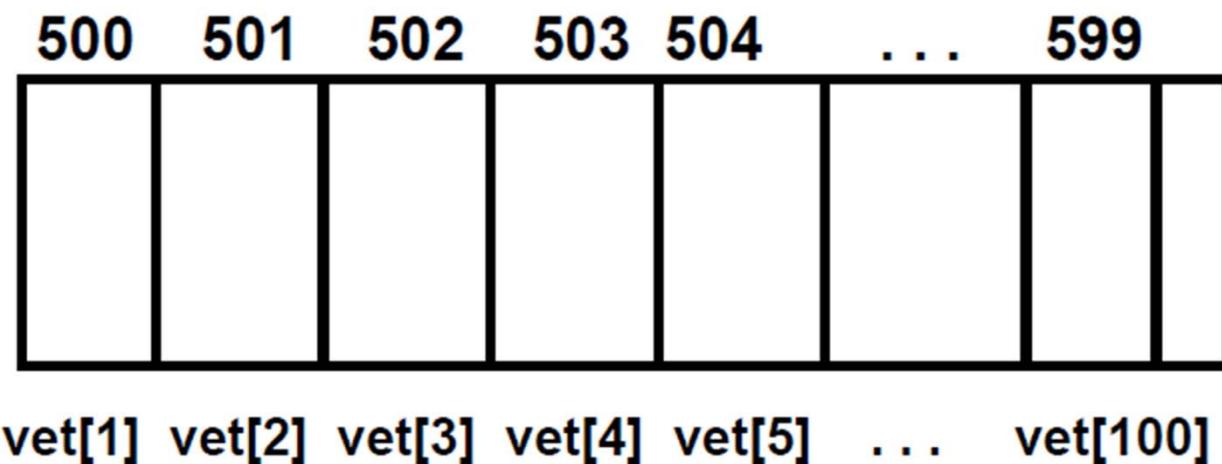
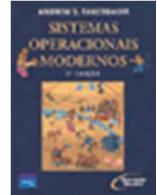


A memória virtual utiliza uma abstração semelhante, só que em relação aos endereços dos programas e aos seus dados. Um programa no ambiente de memória virtual não faz referência a endereços físicos de memória (endereço reais), mas apenas a endereço virtuais. No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço físico, pois o processador acessa apenas posições da memória principal. O mecanismo de tradução do endereço virtual para endereço físico é denominado [mapeamento](#).

O [espaço de endereçamento virtual](#) não tem relação direta com os [endereços no espaço real](#). Um programa pode fazer referência a endereços virtuais que estejam fora dos limites do espaço real, ou seja, os programas e suas estruturas de dados não estão mais limitados ao tamanho da memória física, mas apenas parte deles pode estar residente na memória como extensão da memória principal. Quando um programa é executado só uma parte do código fica residente na memória principal, permanecendo o restante na memória secundária até o momento de ser referenciado.

Quando o usuário desenvolve suas aplicações, ele ignora a existência dos endereços virtuais. [Os compiladores e linkers](#) se encarregam de gerar código executável em função desses endereços, e o [sistema operacional](#) cuida dos detalhes de sua execução.

Gerenciamento de Memória Com Swapping Memória Virtual



Gerenciamento de Memória Com Swapping Mapeamento

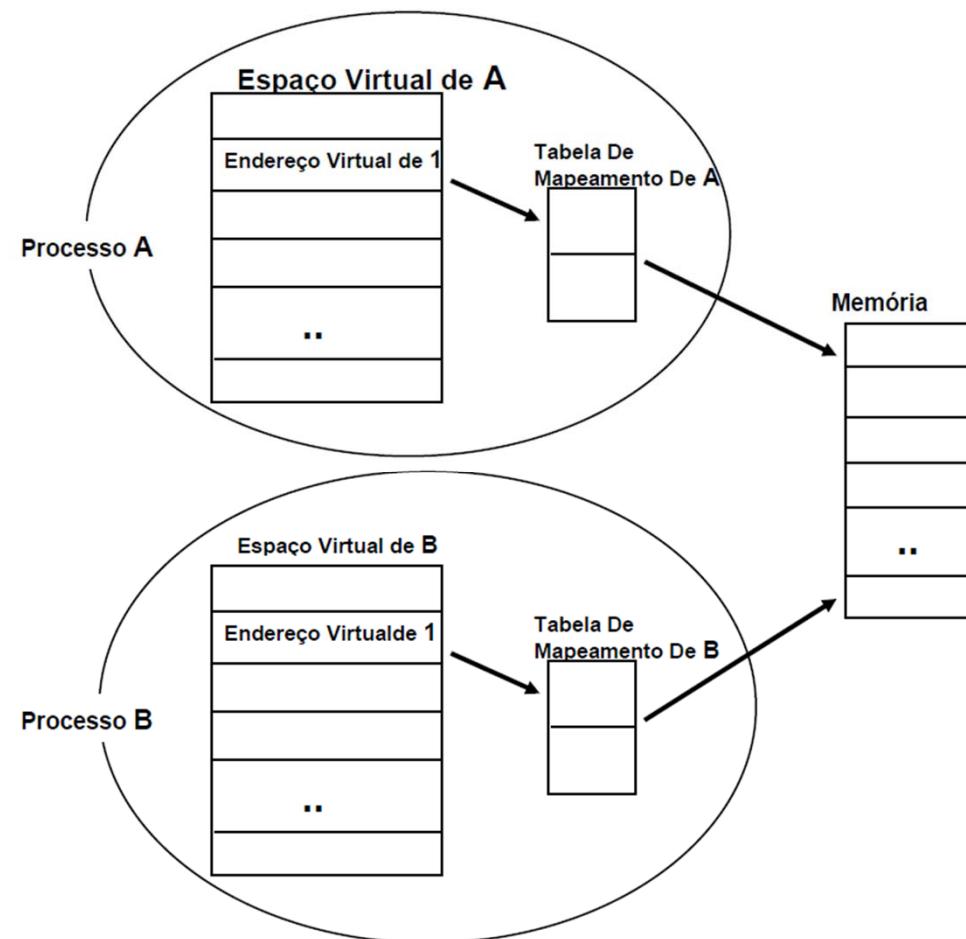


O mecanismo de mapeamento permite ao sistema operacional traduzir um endereço localizado no espaço virtual para um no espaço real, pois o programa executado em seu contexto precisa estar no espaço de endereçamento real para poder ser referenciado ou executado. Portanto um programa não precisa estar necessariamente contíguo na memória real para ser executado.

Cada programa (processo) tem o mesmo espaço de endereçamento virtual, como se possuísse sua própria memória virtual. O mecanismo de tradução se encarrega, então, de manter tabelas de mapeamento exclusivas para cada processo, relacionando os endereços virtuais do processo às suas posições na memória física

Quando um programa está sendo executado, o sistema para realizar a tradução, utiliza a tabela de mapeamento do processo no qual o programa executa. Se um outro programa vai ser executado no contexto de outro processo, o sistema deve passar a referenciar a tabela do novo processo. Isto é realizado através de um registrador, que indica a posição inicial da tabela de mapeamento corrente, onde, toda vez que há mudança de contexto, o registrador é atualizado com o endereço da nova tabela.

Gerenciamento de Memória Com Swapping Mapeamento



Gerenciamento de Memória Com Swapping Paginação



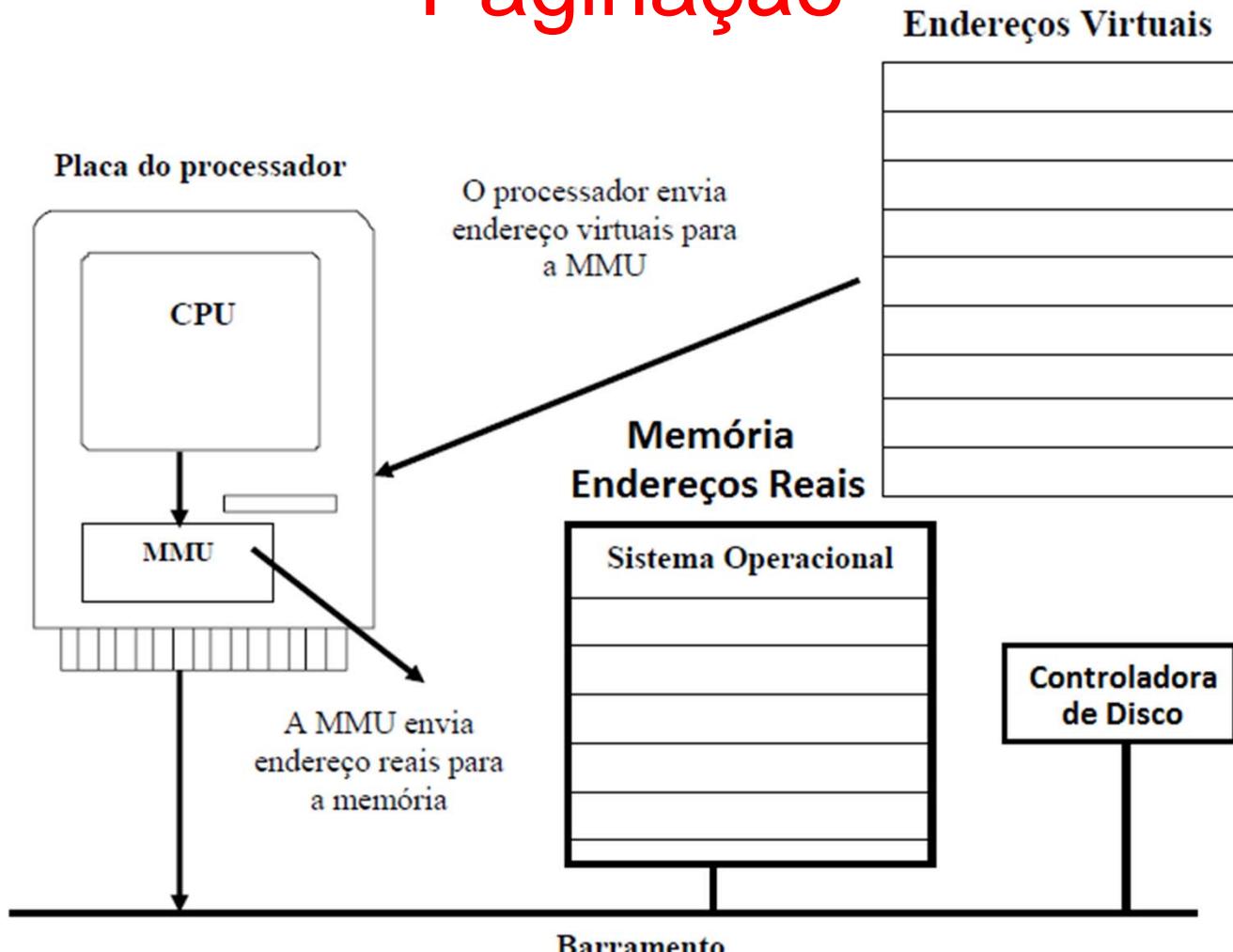
Paginação é a técnica de gerência de memória onde o espaço de endereçamento virtual e o espaço de endereçamento real são divididos em blocos do mesmo tamanho, chamados páginas. No espaço virtual são denominadas páginas virtuais, enquanto as páginas no espaço real são chamados de páginas reais ou frames (quadro).

Os endereços gerados por um programa em execução, são chamados de endereços virtuais e formam o espaço de endereçamento virtual. Em computadores sem memória virtual, o endereço virtual é colocado diretamente no barramento da memória, fazendo com que o endereço físico da memória, equivalente a ele, seja lido ou escrito.

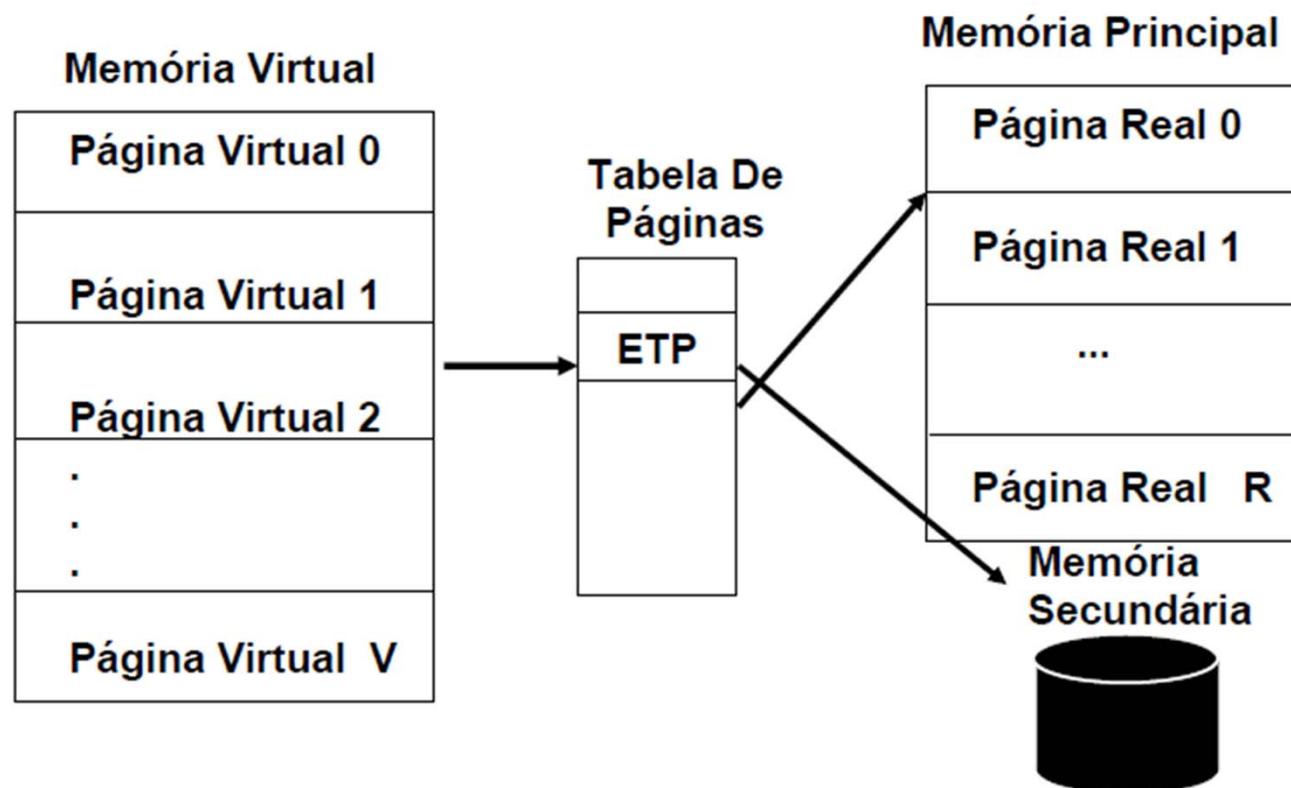
Quando a memória virtual é usada os endereços virtuais não vão direto para o barramento da memória. Em vez disso, eles são encaminhados à unidade de gerência de memória MMU, um chip ou conjunto de chips que mapeiam os endereços virtuais em endereços físicos da memória.

Quando um programa é executado, as páginas virtuais são transferidas da memória secundária para a memória principal e colocadas em frames. Sempre que o programa fizer referência a um endereço virtual, o mecanismo de mapeamento localizará na tabela de mapeamento, o endereço físico do frame.

Gerenciamento de Memória Com Swapping Paginação

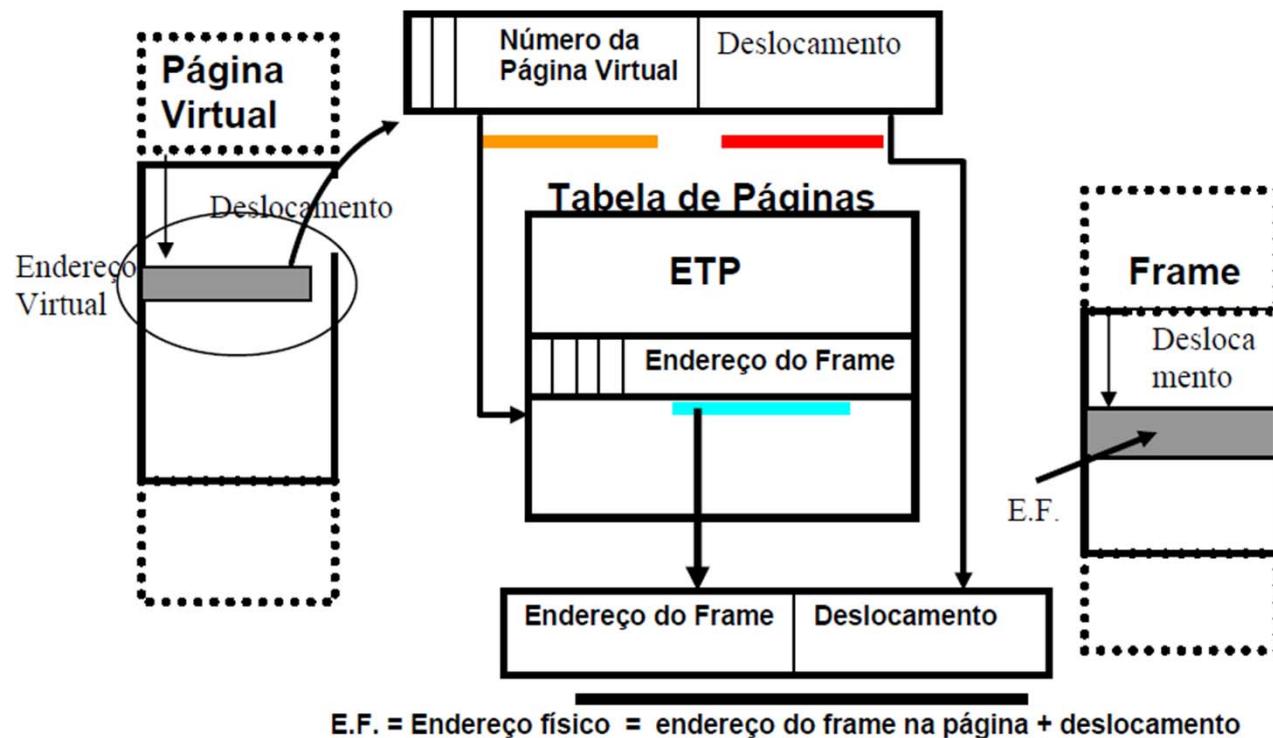


Gerenciamento de Memória Com Swapping Paginação

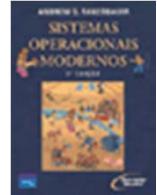


Gerenciamento de Memória Com Swapping Paginação

O endereço virtual é formado pelo número da página virtual e um deslocamento dentro da página. O endereço físico é calculado, então, somando-se o endereço do frame localizado na tabela de páginas com o deslocamento contido no endereço virtual



Gerenciamento de Memória Com Swapping Paginação



Tipos de Paginação

A transferência das páginas de um processo da memória secundária para a principal pode ser:

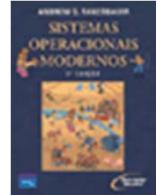
. Apenas quando referenciadas:

Paginação por demanda (demand paging).- apenas as partes do programa realmente referenciadas são trazidas para a memória principal.

. Quando o sistema tenta prever que páginas serão utilizadas :

Paginação anticipada (anticipatory paging).- pode reduzir o overhead na execução de programas. Se o sistema “errar na previsão” recursos desnecessários terão sido gastos.

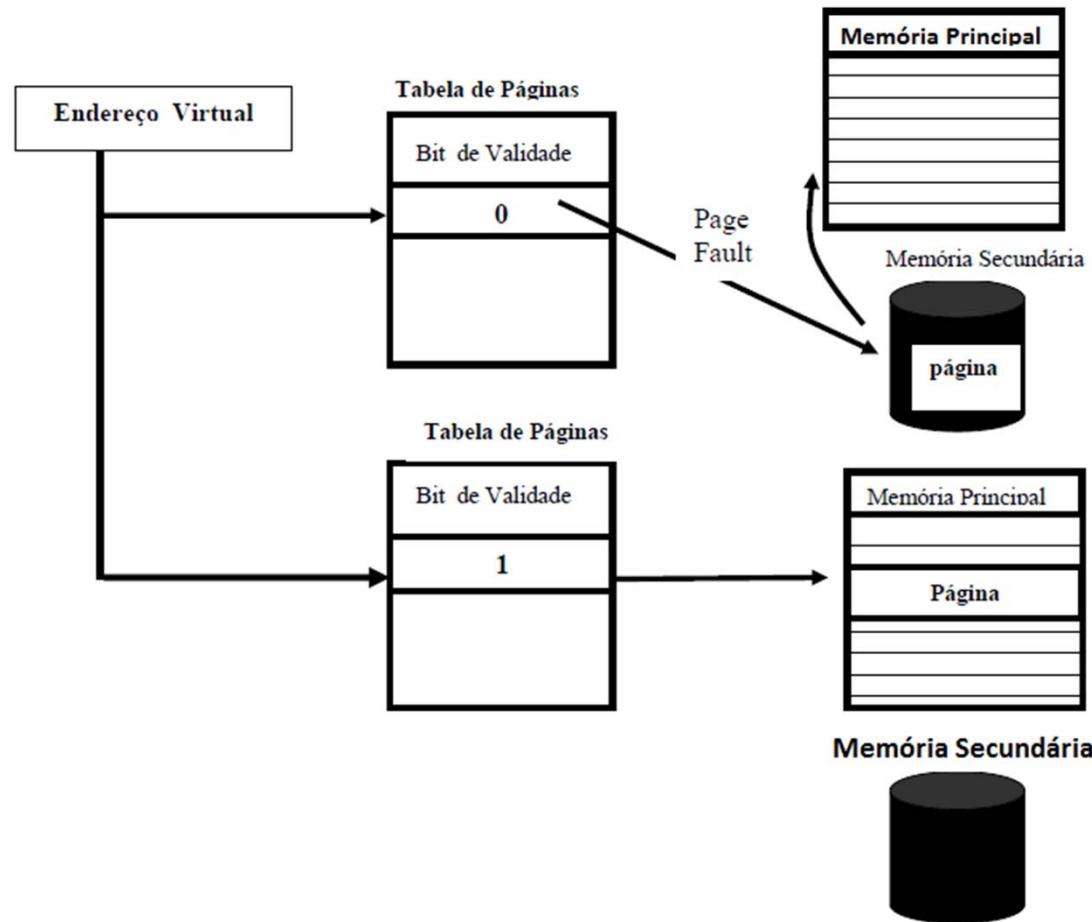
Gerenciamento de Memória Com Swapping Paginação



Bit de Validade

Além da informação sobre a localização da página virtual a entrada na tabela de páginas possui a informação se a página que contém o endereço referenciado está ou não na memória principal, por meio de um bit de validade. Se o bit tem o valor 0, indica que a página virtual não está na memória principal, enquanto, se for igual a 1, a página está localizada na memória. O sistema sempre verifica se a página que contém o endereço referenciado está ou não na memória principal. Caso não esteja, o sistema tem de transferir a página da memória secundária para a memória física.

Gerenciamento de Memória Com Swapping Paginação



Gerenciamento de Memória Com Swapping Paginação



Relocação de Páginas (Troca de páginas)

Quando uma falta de página ocorre, o sistema operacional tem que escolher uma página para remover da memória a fim de dar lugar à que será trazida do disco. Se a página a ser removida foi modificada enquanto estava na memória, ela deve ser reescrita no disco para manter a cópia em disco atualizada. Se, todavia, a página não tiver sido alterada, a cópia do disco já está atualizada, não sendo necessária a reescrita. A página a ser lida é simplesmente superposta à página retirada.

Apesar de ser possível escolher uma página aleatória para dar lugar à página em demanda, o desempenho do sistema é melhorado se for escolhida uma página pouco usada (referenciada). Se uma página muito usada é removida, ela provavelmente terá de ser trazida de volta em breve, resultando um esforço adicional. Algoritmos eficientes de troca de páginas visam minimizar este esforço.

Problema da memória virtual, que páginas remover quando o working set (conjunto de páginas que um processo constantemente referencia e deve permanecer na memória principal do contrário a taxa de paginação pode aumentar) está no limite e precisa de novos frames. O sistema de gerência de memória deve decidir que páginas retirar. Qualquer estratégia deve considerar, se a página selecionada não foi modificada, cuidar para que os dados não sejam perdidos. Para liberar o frame ocupado por uma página, o sistema operacional grava a página na memória secundária (page-out) e mantém um arquivo de paginação (paging file) para páginas modificadas. Sempre que uma página armazenada no arquivo de paginação é referenciada ela é trazida novamente para o working set (page in)

Quando o processo é criado todas as páginas estão na memória secundária e à medida que as referências acontecem, as páginas são transferidas (page in) para o working set (na memória principal). O sistema, agora, verifica antes se a página referenciada está no working set.

Gerenciamento de Memória Com Swapping Paginação



Algoritmos de Troca de Páginas

- **Aleatória (random)**

- escolhe uma página qualquer do working set
- consome poucos recursos
- raramente utilizada

- **First-in-first-out (FIFO)**

- a página primeiro utilizada sairá primeiro do working set
- fila FIFO
- critério razoável ?
 - » e o caso de loops (página constantemente referenciada) ?
 - » e o caso de utilitários do sistema operacional (que não ficam no núcleo) que são constantemente utilizados ?
- pode acarretar muitos page faults

- **Least-recently-used (LRU)**

- seleciona página usada menos recentemente (página que está há mais tempo sem ser referenciada)
- boa estratégia
- muito overhead
 - » atualização, em cada página referenciada
 - » do momento do último acesso
 - » busca destas páginas (ou ordenação a cada acesso ?)

Gerenciamento de Memória Com Swapping Paginação



- **Not-recently-used (NRU)**

- página não usada recentemente (similar ao LRU)
- implementação
 - » um flag indica quando a página foi referenciada ou não
 - » flag associado a cada entrada na tabela de páginas
 - » inicialmente todas as páginas estão com o flag indicando que ainda não foram acessadas
- à medida que forem sendo referenciadas o flag é modificado pelo HW
- depois de um certo tempo pode-se saber que páginas não foram referenciadas

- **Least-frequently-used (LFU)**

- a página menos referenciada é selecionada (menos frequentemente usada)
- é mantido um contador com o número de referências feitas às páginas
- a que tiver menor contagem é selecionada
- este algoritmo privilegia páginas bastante utilizadas
- parece ser uma boa estratégia
 - »entretanto, páginas entram mais recentemente no working set são, justamente, as escolhidas pois estão com os contadores em menor valor



Gerenciamento de Memória Com Swapping Segmentação

Segmentação é a técnica de gerência de memória, onde os programas são divididos logicamente em sub-rotinas e estruturas de dados e colocados em blocos de informações na memória. Os blocos têm tamanhos diferentes e são chamados *segmentos*, cada um com seu próprio espaço de endereçamento.

Memória Principal

```

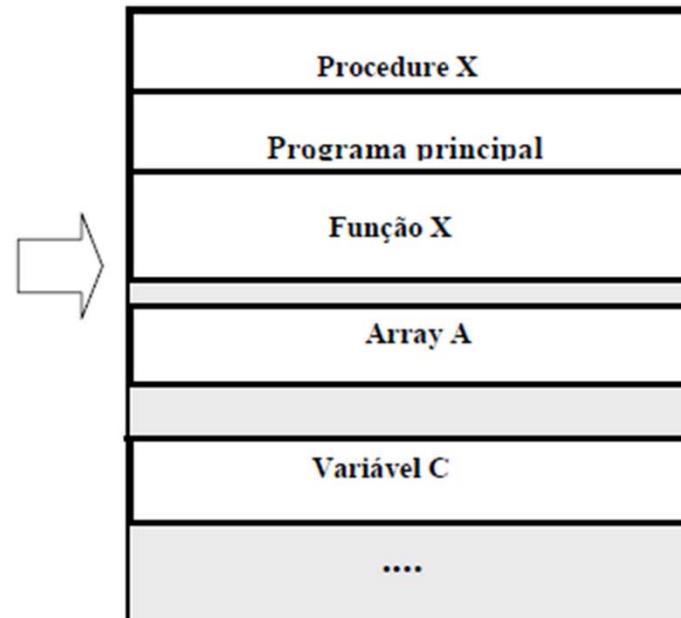
PROGRAM segmento
Var A : array ...
C : integer;

PROCEDURE X;
.

END;

FUNCTION Y;
.

END;
BEGIN
.
} Programa principal
.
END.
  
```



Gerenciamento de Memória Com Swapping Segmentação



A grande diferença entre a paginação e a segmentação é que, enquanto a primeira divide o programa em partes de tamanho fixo, sem qualquer ligação com a estrutura do programa, a segmentação permite uma relação entre a lógica do programa e sua divisão na memória.

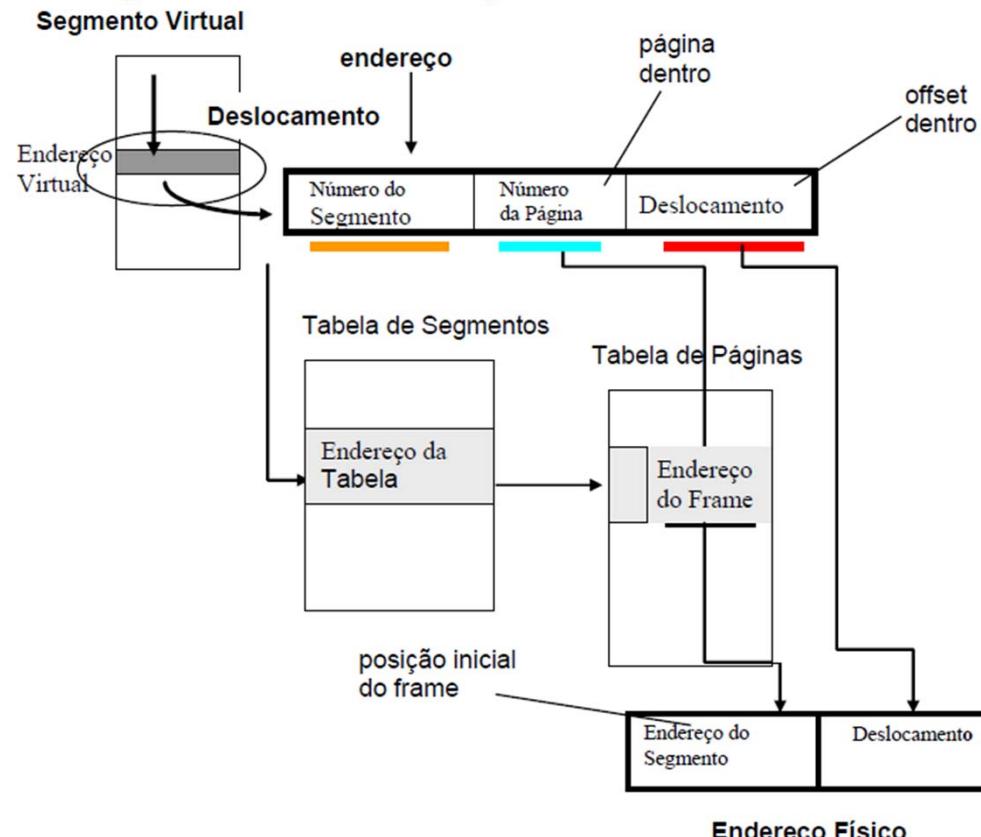
O mecanismo de mapeamento é muito semelhante ao da paginação. Além do endereço do segmento na memória física cada entrada na tabela de segmentos possui informações sobre o tamanho do segmento, se ele está ou não na memória.

Se as aplicações não estiverem divididas em módulos, grandes pedaços de código estarão na memória desnecessariamente, não permitindo que outros usuários, também utilizem a memória. O problema da fragmentação também ocorre nesse modelo, quando as áreas livres são tão pequenas, que não acomodam nenhum segmento que necessite ser carregado.

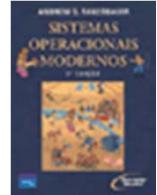
Gerenciamento de Memória Com Swapping Segmentação com Paginação



Sistemas que implementam segmentação com paginação, permitem a divisão lógica dos programas em segmentos e, por sua vez, cada segmento é dividido, fisicamente em páginas.



Gerenciamento de Memória Com Swapping Compartilhamento de Memória



Em sistemas multiprogramáveis, é comum usuários utilizarem certos programas simultaneamente (código reentrante), o que evita que várias cópias de um mesmo programa ocupem a memória desnecessariamente. Um bom exemplo desse tipo de aplicação são os utilitários do sistema, como compiladores, editores de texto, etc...

Em sistemas que implementam memória virtual, é bastante simples o compartilhamento de código e dados entre vários processos. Para isso, basta que as entradas das tabelas de páginas apontem para as mesmas páginas na memória principal. Desta forma, é possível reduzir-se o número de programas na memória e aumentar o número de usuários compartilhando o mesmo recurso. Este esquema pode ser visualizado na figura com os ponteiros dos processos A e B, apontando para a mesma área de memória.

Gerenciamento de Memória Com Swapping Compartilhamento de Memória

