Aula 13 – Memória Virtual e Paginação

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

31 de outubro de 2014

Sobrecarga de Memória

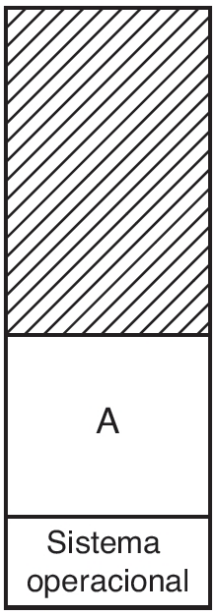
- O que acontece quando a memória disponível não é suficiente para todos os processos?
- Existem 2 métodos gerais para lidar com a sobrecarga de memória:
  - A troca de processos (swapping):
    - Consiste em trazer, em sua totalidade, cada processo para a memória, executá-lo durante um tempo e, então, devolvê-lo ao disco
    - Assim, processos ociosos acabam ficando no disco
  - Memória Virtual:
    - Permite que programas possam ser executados mesmo que estejam parcialmente carregados na memória principal

Swapping

- Chaveamento de processos inteiros entre a memória principal e o disco
- Swap-out
  - Da memória para uma região especial do disco, chamada "área de swap"
- Swap-in
  - Do disco pra memória

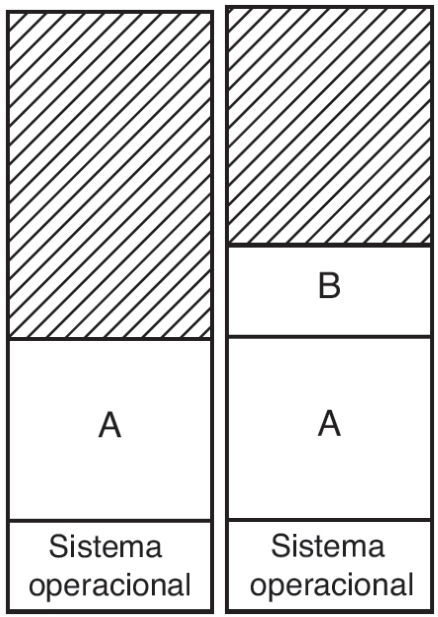
Swapping

- Funcionamento:
  - Inicialmente o processo A está na memória



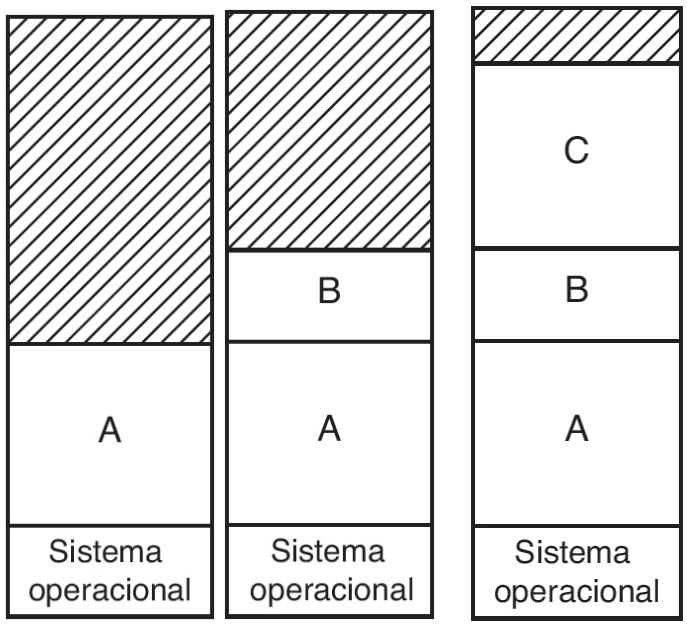
Swapping

- Funcionamento:
  - Inicialmente o processo A está na memória
  - Então B e C são criados ou trazidos do disco (swap-in)



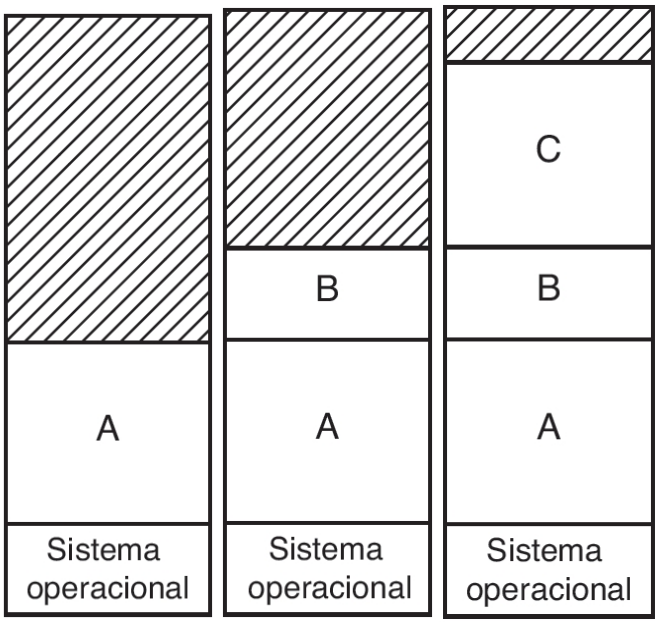
Swapping

- Funcionamento:
  - Inicialmente o processo A está na memória
  - Então B e C são criados ou trazidos do disco (swap-in)



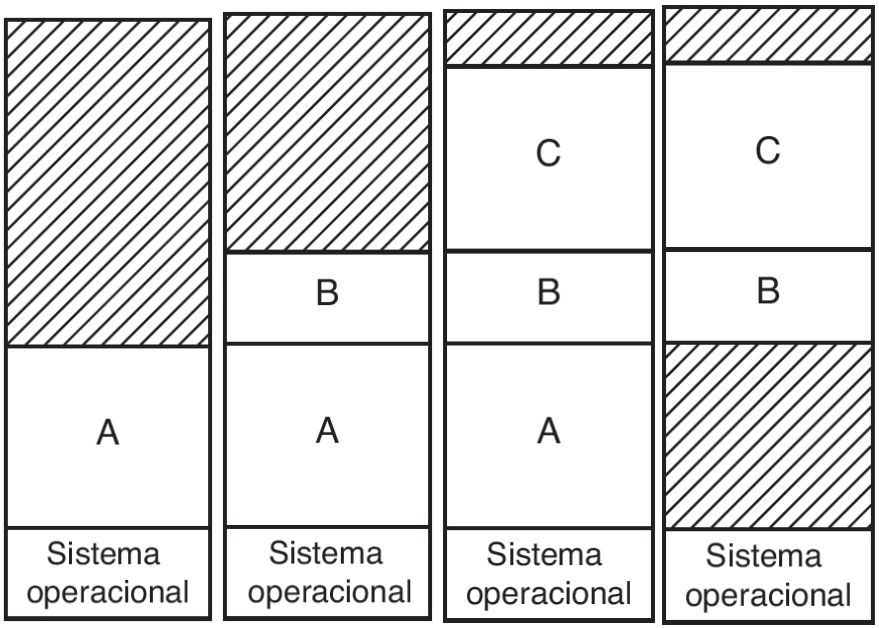
Sobrecarga de Memória – Swapping

- Funcionamento:
  - D é iniciado. O processo A é devolvido ao disco (swap-out), dando espaço a D, que entra na memória



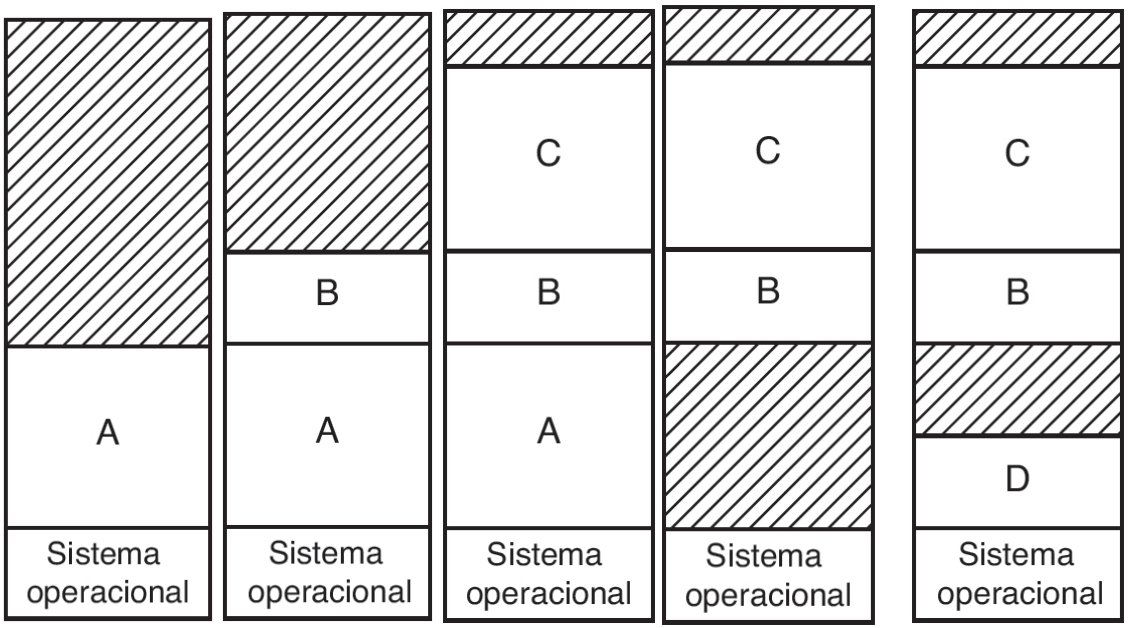
Sobrecarga de Memória – Swapping

- Funcionamento:
  - D é iniciado. O processo A é devolvido ao disco (swap-out), dando espaço a D, que entra na memória



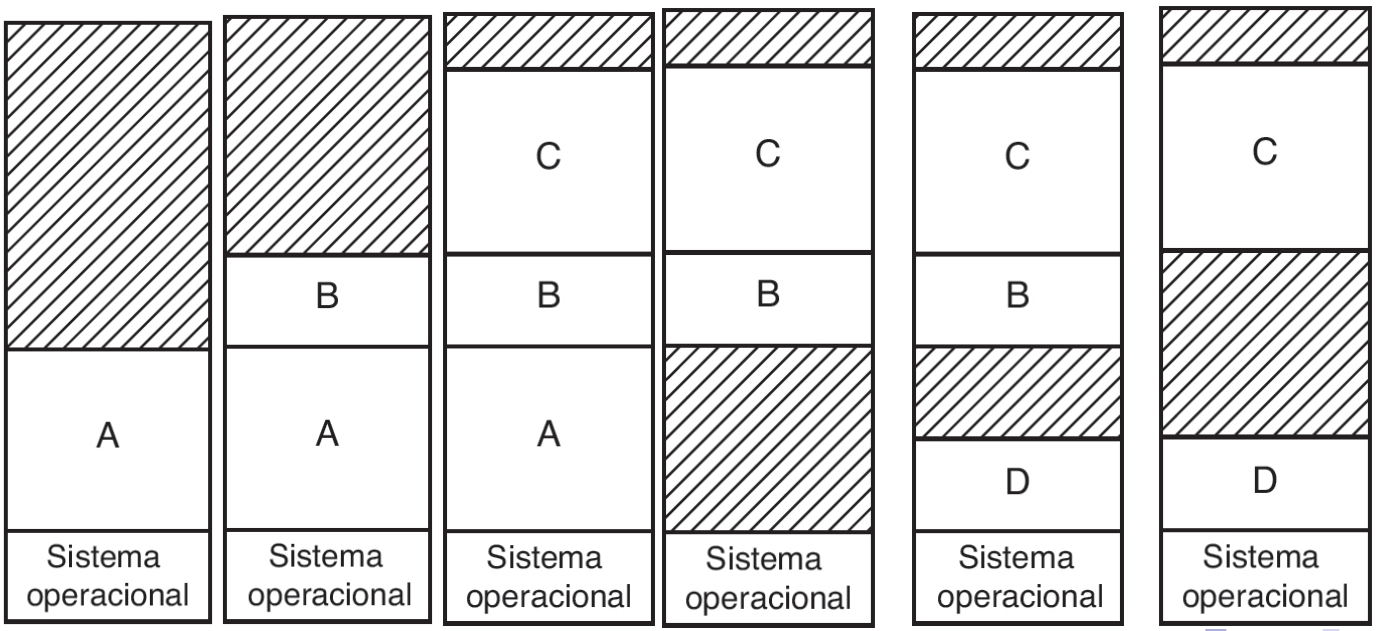
Sobrecarga de Memória – Swapping

- Funcionamento:
  - D é iniciado. O processo A é devolvido ao disco (swap-out), dando espaço a D, que entra na memória



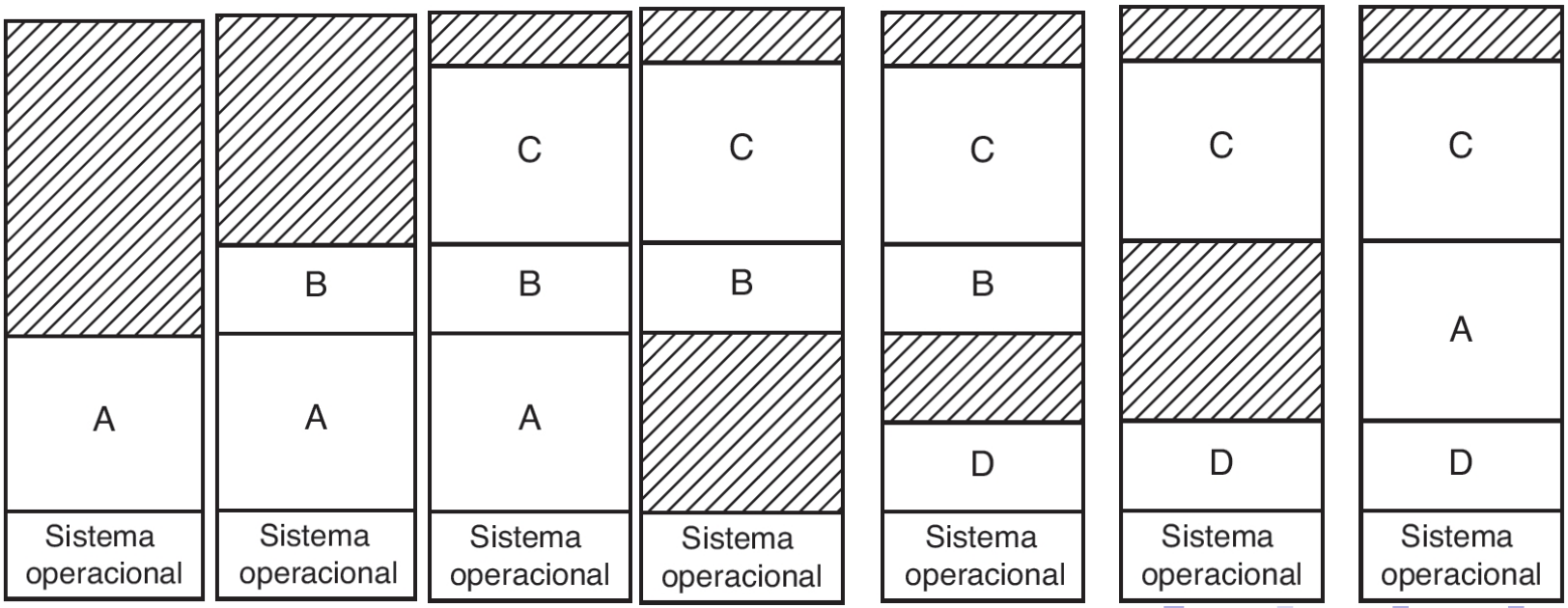
Sobrecarga de Memória – Swapping

- Funcionamento:
  - D é iniciado. O processo A é devolvido ao disco (swap-out), dando espaço a D, que entra na memória
  - A precisa rodar novamente. B é retirado,



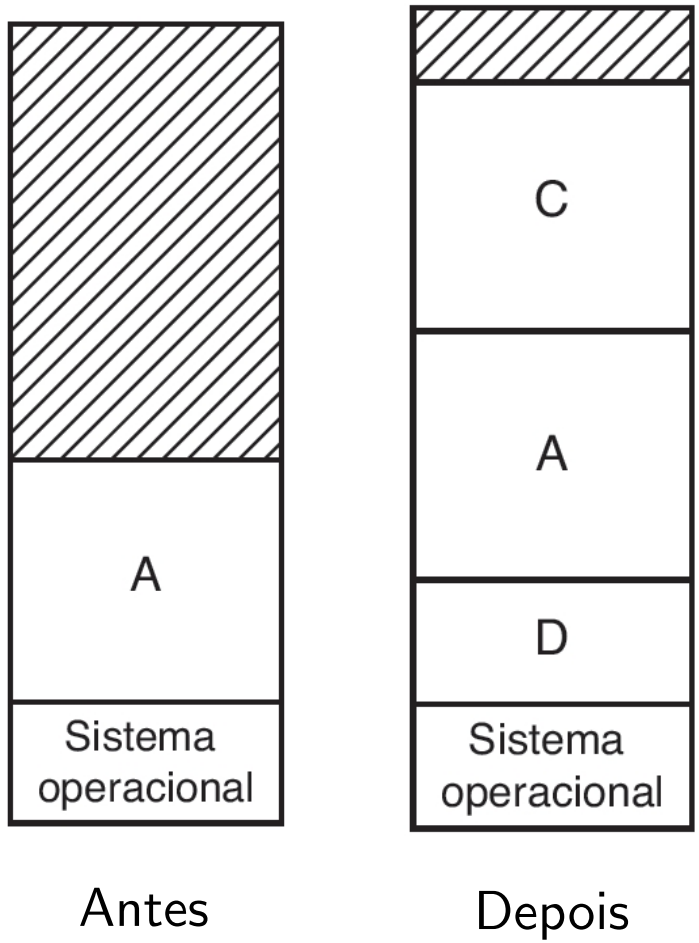
Sobrecarga de Memória – Swapping

- Funcionamento:
  - D é iniciado. O processo A é devolvido ao disco (swap-out), dando espaço a D, que entra na memória
  - A precisa rodar novamente. B é retirado, e A é novamente trazido do disco para a memória



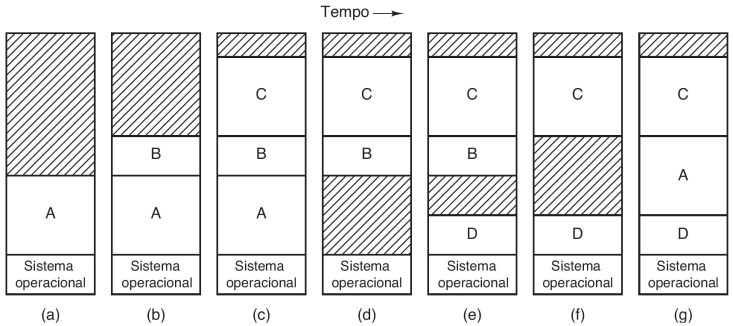
Swapping

- Note que A agora está em outra porção da memória
  - Devemos relocar os endereços em A via software durante a carga na memória
  - Ou via hardware durante a execução do programa
    - Ex: via registradores base e limite



Swapping

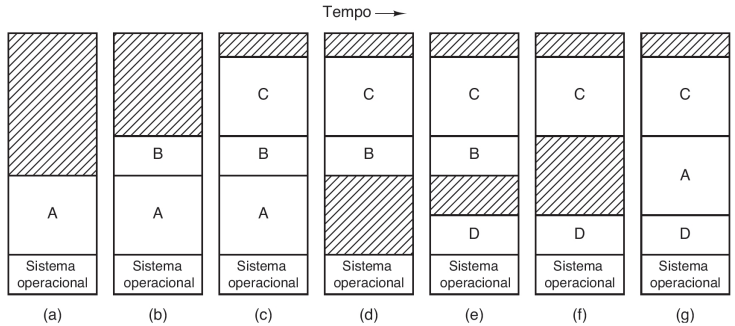
- Problema
  - A operação de swap pode criar muitos “buracos” na memória – fragmentação externa



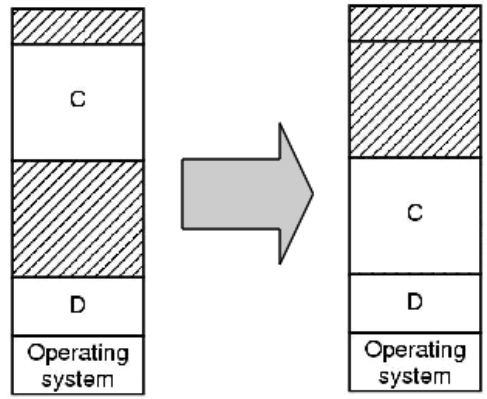
  - Pode ficar difícil de acomodar outro processo no buraco

Swapping

- Problema
  - A operação de swap pode criar muitos “buracos” na memória – fragmentação externa

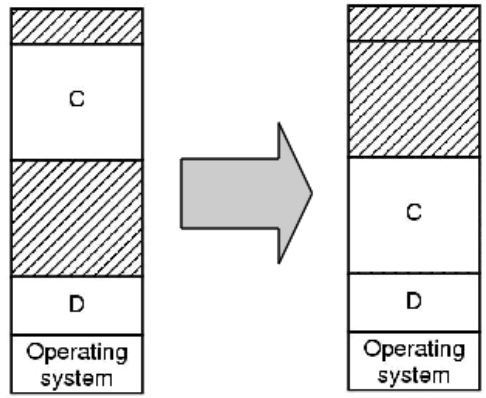


  - Pode ficar difícil de acomodar outro processo no buraco
- Solução: Compactação de memória:
  - Mova todos os processos o mais para baixo possível na memória – haverá um único espaço vazio acima
  - Consome bastante CPU



Swapping

- Compactação de Memória – Ex:
  - Qual o tempo para compactar 1GB?
    - Se copia 4B (32b) em 20ns



$$\frac{1.073.741.824B \times 20ns}{4B} = 5.368.709.120ns$$

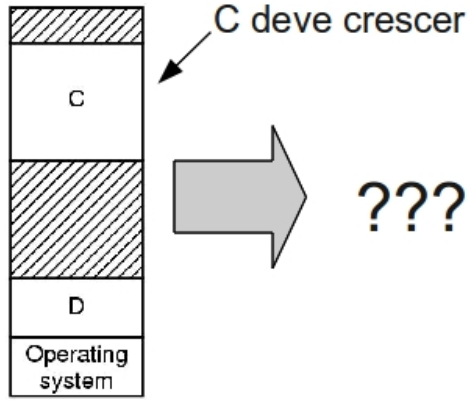
$$\text{ou} \approx 5.4s$$

Swapping – Alocação de Memória

- A alocação de memória muda à medida em que
  - Os processos chegam à memória
  - Os processos deixam a memória
- Quanto de memória devemos alocar a um processo quando ele é criado ou trazido do disco?
  - Se eles tiverem tamanho fixo, aloque o que ele precisa

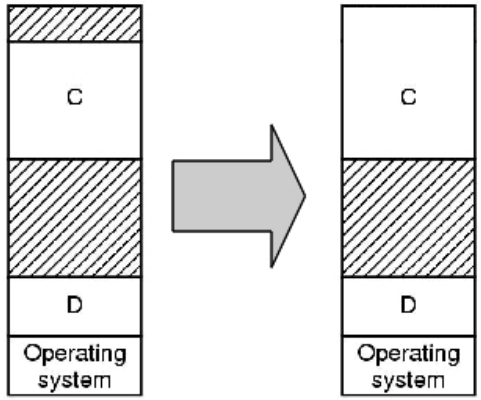
Swapping – Alocação de Memória

- Mas e se o segmento de dados crescer com o tempo?
  - Ex: O processo aloca memória dinamicamente



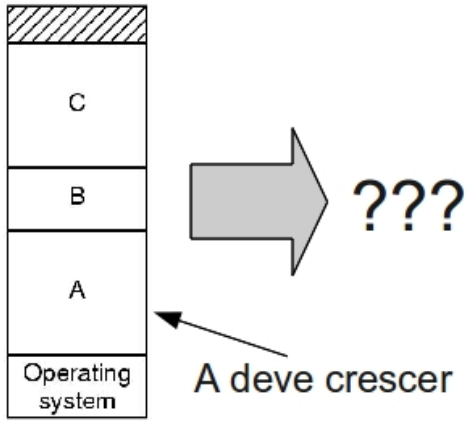
Swapping – Alocação de Memória

- Mas e se o segmento de dados crescer com o tempo?
  - Ex: O processo aloca memória dinamicamente
  - Se houver um "buraco" adjacente à memória atual do processo, ele pode ser alocado, e o processo cresce nesse buraco



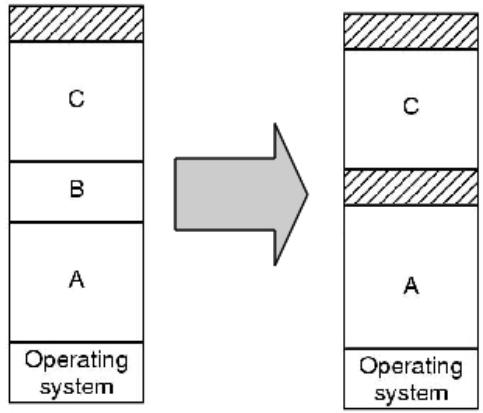
Swapping – Alocação de Memória

- E se ele for adjacente a outro processo?



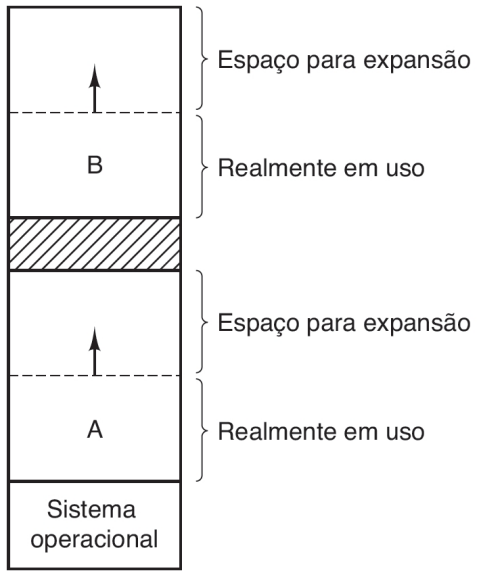
Swapping – Alocação de Memória

- E se ele for adjacente a outro processo?
  - Ou o movemos a um buraco maior na memória
  - Ou um ou mais processos terão que ser transferidos para o disco, para criar esse buraco
  - Se o processo não puder crescer na memória e a área de troca de disco (swap) estiver cheia, ele pode ser suspenso até que algum espaço seja liberado



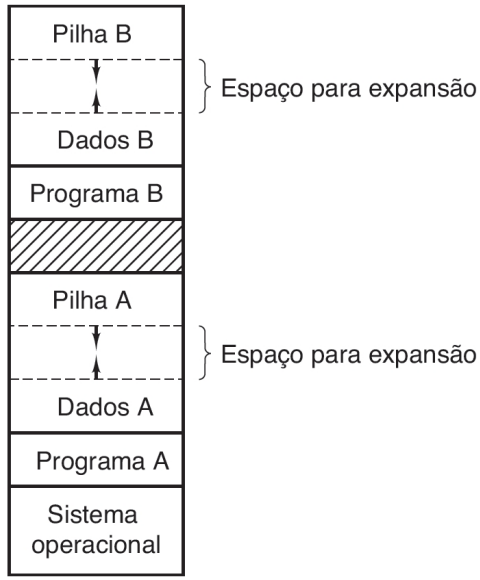
Swapping – Alocação de Memória

- Alternativamente:
  - Se o esperado é que a maioria dos processos cresça durante a execução, podemos alocar uma memória extra quando o processo é carregado
    - Reduz o overhead de ter que fazer swap ou movê-lo na memória
    - Se ainda assim o processo for para o disco, somente a memória realmente em uso é gravada



Swapping – Alocação de Memória

- Se os processos tiverem dois segmentos que crescem (dados e pilha, por exemplo), podemos usar a mesma ideia
  - Nesse caso, a pilha cresce para baixo enquanto que o segmento de dados cresce para cima
  - A porção de memória entre essas duas áreas pode ser usada por ambas



- Se ainda assim ficar sem espaço:
  - Transfere o processo a outro local, faz swap, ou termina o processo

Gerenciando a Memória

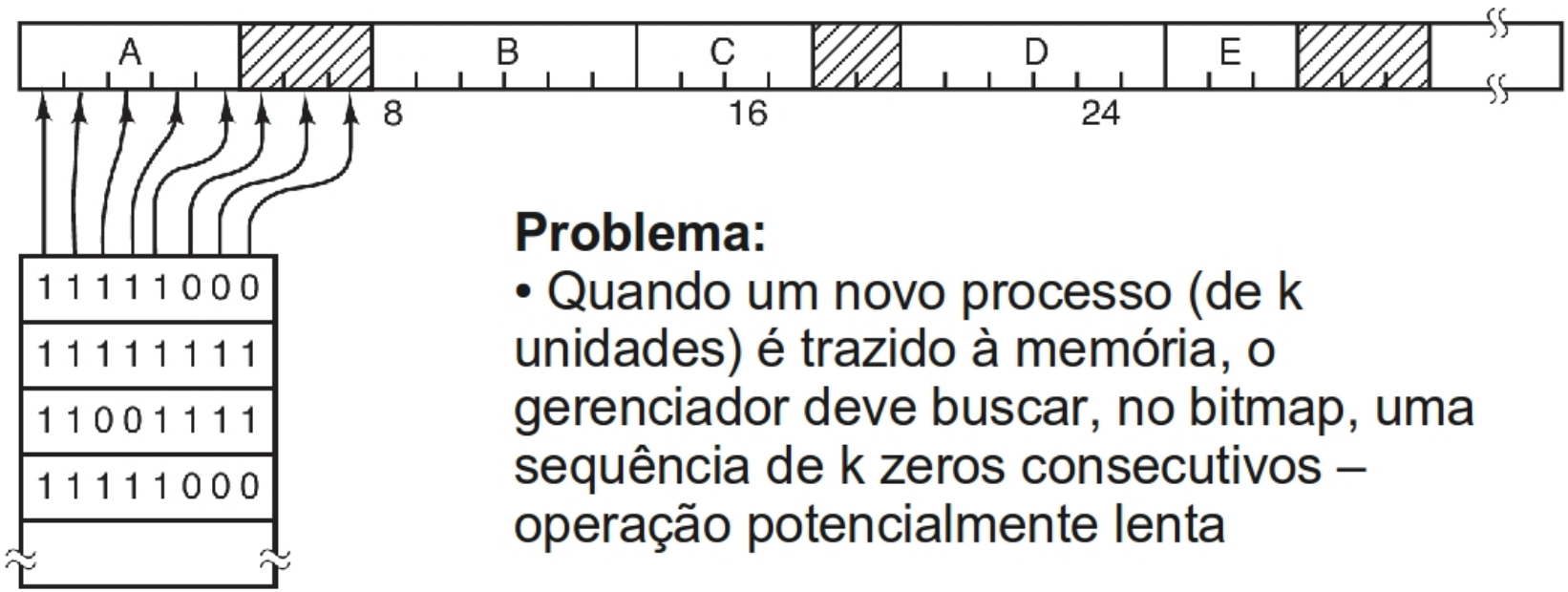
- Quando se atribui memória dinamicamente, o SO deve gerenciá-la
- O gerenciamento pode ser feito por:
  - Mapas de bits
  - Listas ligadas

Gerenciando a Memória: Mapas de Bits

- Memória é dividida em unidades de alocação
  - Pode conter até vários KB
- Cada unidade corresponde a um bit no bitmap:
  - 0 $\rightarrow$ livre
  - 1 $\rightarrow$ ocupado
- Tamanho do bitmap depende do tamanho da unidade e do tamanho da memória
  - unidades de alocação pequenas $\rightarrow$ bitmap grande
  - unidades de alocação grandes $\rightarrow$ perda de espaço

Gerenciando a Memória: Mapas de Bits

- Vantagem:
  - Ocupa um tamanho fixo na memória, pois só depende do tamanho da memória e da unidade de alocação



Problema:

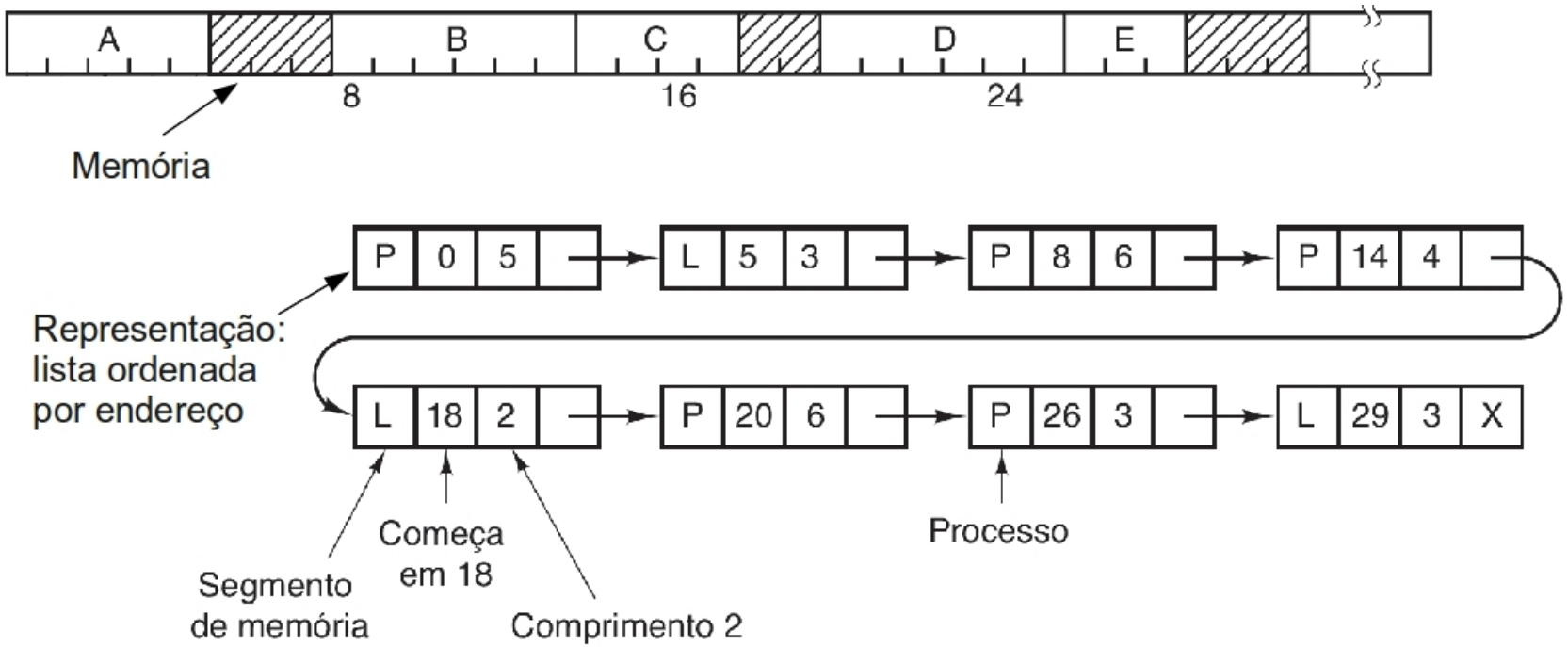
• Quando um novo processo (de k unidades) é trazido à memória, o gerenciador deve buscar, no bitmap, uma sequência de k zeros consecutivos – operação potencialmente lenta

Mapas de Bits – Fragmentação

- Interna
  - Desperdício de memória dentro da área alocada para um processo
  - Ex.: Unidades de alocação grandes e um processo cujo tamanho não é múltiplo da unidade de alocação
- Externa:
  - Desperdício fora da área alocada para um processo
  - Ex: Memória fragmentada após várias trocas com o disco, sem que haja compactação

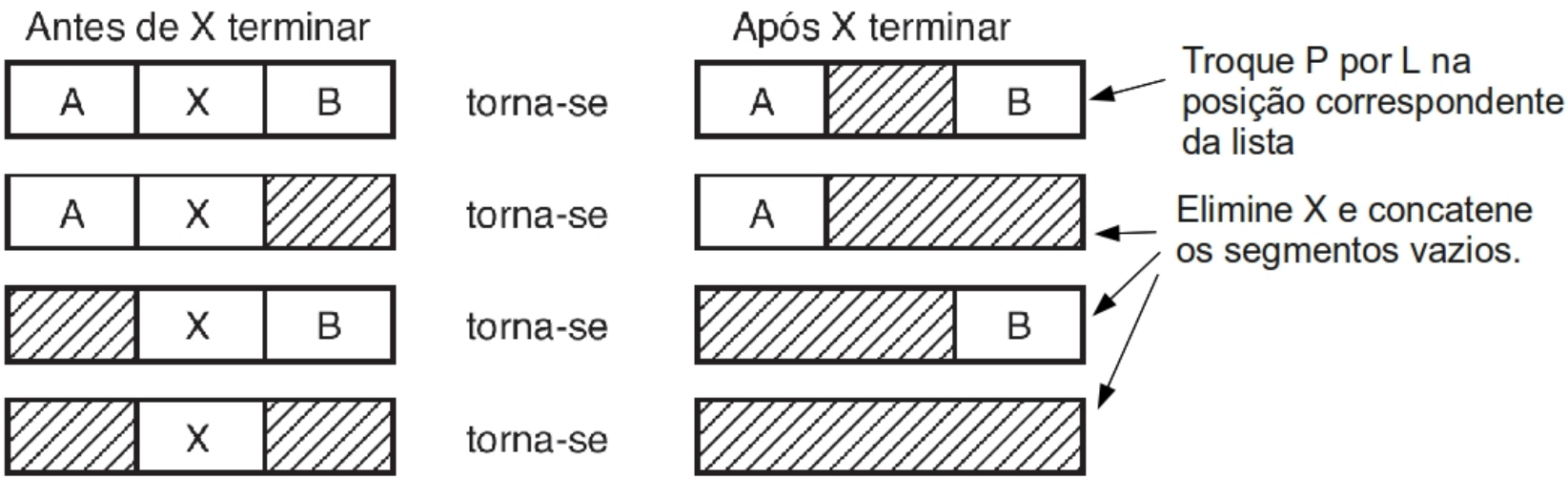
Gerenciando a Memória: Listas Ligadas

- Mantêm-se uma lista ligada de segmentos de memória livres e alocados
  - Cada segmento ou contém um processo ou é um buraco vazio entre dois processos



Gerenciando a Memória: Listas Ligadas

- A lista ordenada por endereço fica fácil de ser atualizada:



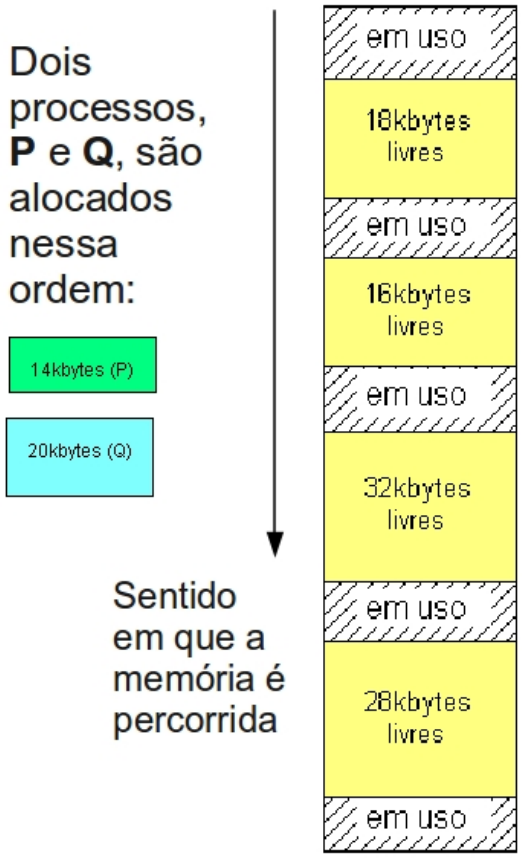
Gerenciando a Memória: Listas Ligadas

- Com uma lista ordenada por endereço, como alocar memória a um processo?
  - Supondo que o gerenciador de memória saiba quanta memória deve ser alocada ao processo
- Primeiro encaixe (First fit):
  - O gerenciador percorre a lista de segmentos, desde o início, até que encontre um buraco grande o bastante
    - Quebra o buraco em dois pedaços – um para o processo e um para a memória ainda livre
  - Variação: inicie a busca a partir de onde parou na vez anterior (**next fit** – próximo encaixe)
    - Deve memorizar a última posição em que encontrou memória disponível suficientemente grande

Gerenciando a Memória: Listas Ligadas

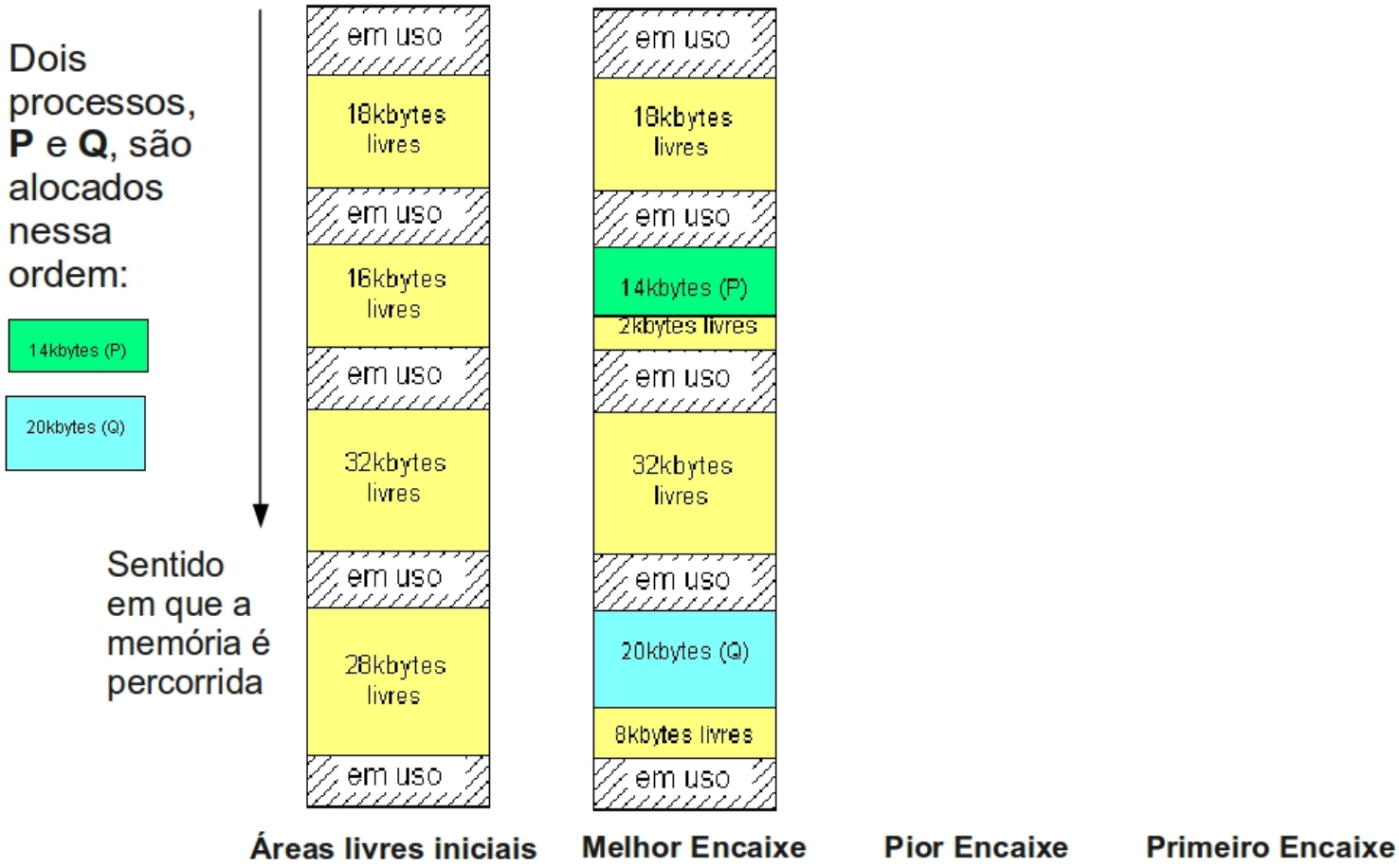
- Melhor encaixe (Best fit):
  - Busca a lista inteira, até o fim, e toma o menor buraco que seja adequado
  - Não quebra buracos grandes que poderiam ser úteis mais tarde
  - Problema: permite o surgimento de buracos minúsculos
- Pior encaixe (worst fit):
  - Sempre tome o maior buraco disponível
  - Reduz a chance de buracos minúsculos e inúteis

Gerenciando a Memória: Listas Ligadas

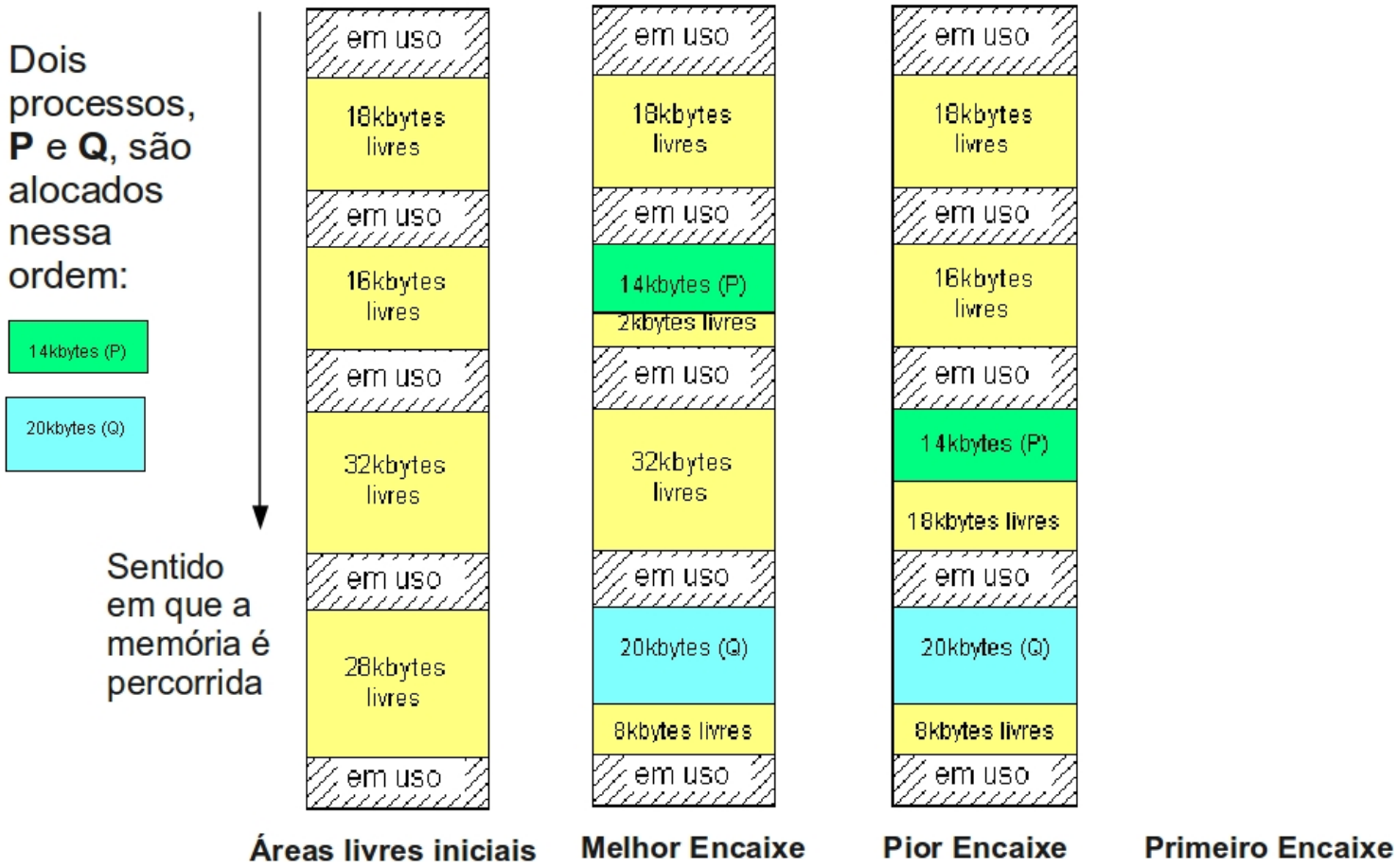


Áreas livres iniciais **Melhor Encaixe** **Pior Encaixe** **Primeiro Encaixe**

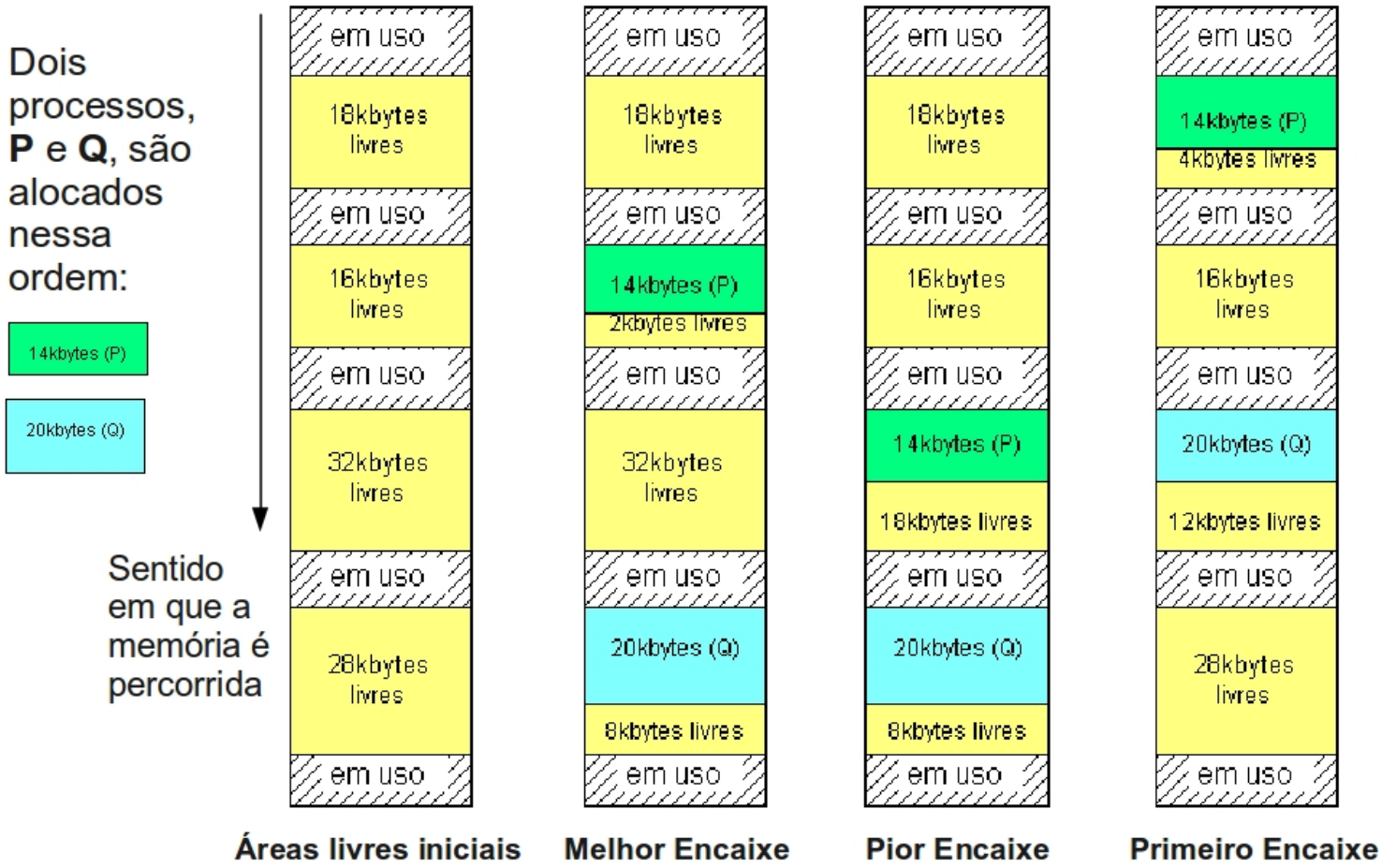
Gerenciando a Memória: Listas Ligadas



Gerenciando a Memória: Listas Ligadas



Gerenciando a Memória: Listas Ligadas



Gerenciando a Memória: Listas Ligadas

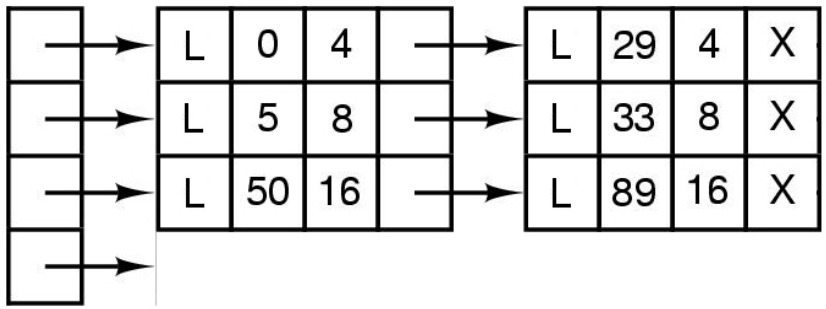
- Principais consequências dos algoritmos:
  - Melhor encaixe: deixa o menor resto, porém após um longo processamento poderá deixar "buracos" muito pequenos para serem úteis
  - Pior encaixe: deixa o maior espaço após cada alocação, mas tende a espalhar as porções não utilizadas sobre áreas não contínuas de memória e, portanto, pode tornar difícil alocar grandes processos
  - Primeiro encaixe: tende a ser um meio termo entre a melhor e a pior escolha, com a característica adicional de fazer com que os espaços vazios migrem para o final da memória. Na prática, uma boa escolha.

Gerenciando a Memória: Listas Ligadas

- Esses algoritmos poderiam ser mais rápidos se segmentos de memória em uso e livres fossem mantidos em listas separadas
  - A busca se daria somente na lista de segmentos livres
  - Se ordenada por tamanho, a lista de livres tornaria o *best fit* e *worst fit* mais rápidos
- Problema
  - Liberar memória fica mais lento e complicado
    - Deve-se remover da lista de segmentos em uso e incluir na de livres

Gerenciando a Memória: Listas Ligadas

- Encaixe rápido (Quick fit)
  - Mantém listas separadas para alguns dos tamanhos de memória livre mais comumente requisitados
    - Uma para cada tamanho
    - A busca por um segmento livre de um determinado tamanho é rápida
    - Tamanhos incomuns são incluidos em uma lista à parte, ou na lista existente mais próxima
  - Problema quando processo é desalocado:
    - Encontrar os vizinhos ao buraco que ele deixou, para união



Memória Virtual

Memória Virtual

- Registradores base e limite criam a abstração de espaço de endereço
- Swap trata do problema de não termos memória suficiente para todos os processos
  - Exige que haja memória para cada processo individualmente
- E se um único processo ocupar mais memória que o disponível (um *bloatware*)?
  - Hoje em dia é comum necessidade de rodar programas grandes demais para a memória

Sobrecarga de Memória – Memória Virtual

- Que fazer?
- Memória virtual (Fotheringham, 1961)
  - Método para alocação de processos na memória
    - Continuamos "enganando" o processo, quanto a que endereço de memória ele está realmente usando
  - Usa duas técnicas principais (normalmente juntas):
    - Paginação – blocos de tamanho fixo
    - Segmentação – blocos de tamanho variável

Memória Virtual – Paginação

- A ideia básica é que cada programa tenha seu próprio espaço de endereçamento, que é dividido em blocos de tamanho fixo chamados páginas
  - Cada página é uma série contígua de endereços, mapeada na memória física
  - Nem todas precisam estar na memória física para executar o programa
  - Quando o programa referencia um endereço em uma página que está na memória, o hardware executa o mapeamento necessário dinamicamente

Memória Virtual – Paginação

- Quando o programa referencia um endereço em uma página que não está na memória
  - A instrução falha – page fault (trap)
  - O SO deve obter a página que falta e reexecutar a instrução que falhou
    - O processo recebe memória física sempre que houver memória disponível
- Com multiprogramação, a memória contém partes de diferentes programas
  - Se um programa estiver esperando por outra página, a CPU pode ser escalonada a outro processo

Real × Virtual

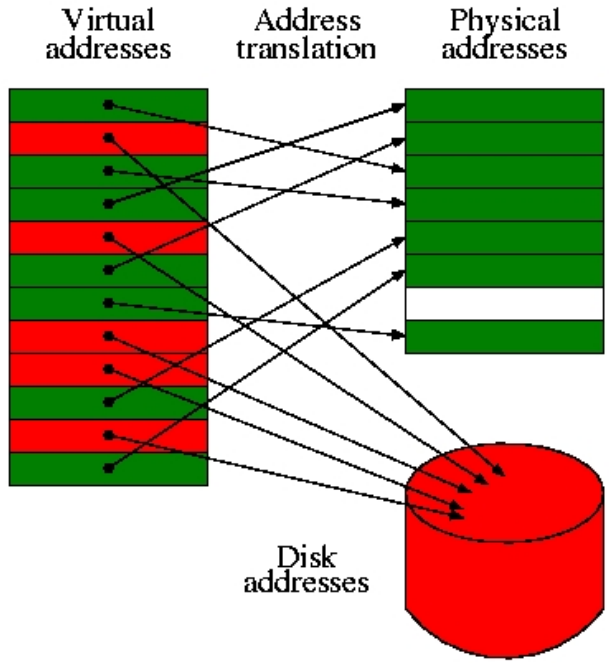
- Os endereços físicos (reais) podem ser gerados de muitas maneiras:
  - Mapeamento direto
  - Usando registradores base e limite
  - etc
  - O endereço resultante é colocado diretamente no barramento de memória
- Espaço de Endereçamento Físico de um processo:
  - Formado por todos os endereços físicos/reais aceitos pela memória principal (RAM)

Real × Virtual

- Endereços gerados pelos programas são chamados de virtuais ou lógicos
  - Na ausência de memória virtual, o endereço virtual é idêntico ao físico
  - Com memória virtual, é mapeado à memória física (não vai diretamente ao barramento)
- Espaço de Endereçamento Virtual de um processo:
  - Formado por todos os endereços virtuais que esse processo pode gerar
  - Não necessariamente correspondem diretamente aos físicos
  - O espaço de endereço físico de um processo pode ser não contíguo

Memória Virtual – Paginação

- Um processo em Memória Virtual faz referência a endereços virtuais e não a endereço reais de RAM
- No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço real
  - A CPU manipula apenas endereços reais da RAM
  - Deve haver um mapeamento



Memória Virtual – Paginação

- E como se dá o mapeamento?
  - Tabela de páginas: responsável por armazenar informações sobre as páginas virtuais:
    - argumento de entrada $\rightarrow$ número da página virtual
    - argumento de saída (resultado) $\rightarrow$ número da página real (ou moldura de página)
  - Gerenciada pela MMU (já já...)
- As transferências entre memória e disco são sempre em páginas completas
  - Página é a unidade básica para transferência de informação

Memória Virtual – Paginação

- E quem faz esse mapeamento?
  - Um hardware especial, presente na CPU → MMU (memory management unit)