



Universidade Federal
de Mato Grosso
Campus Rondonópolis

Sistemas Operacionais

Aula 11 - Gerenciamento de Memória (1)

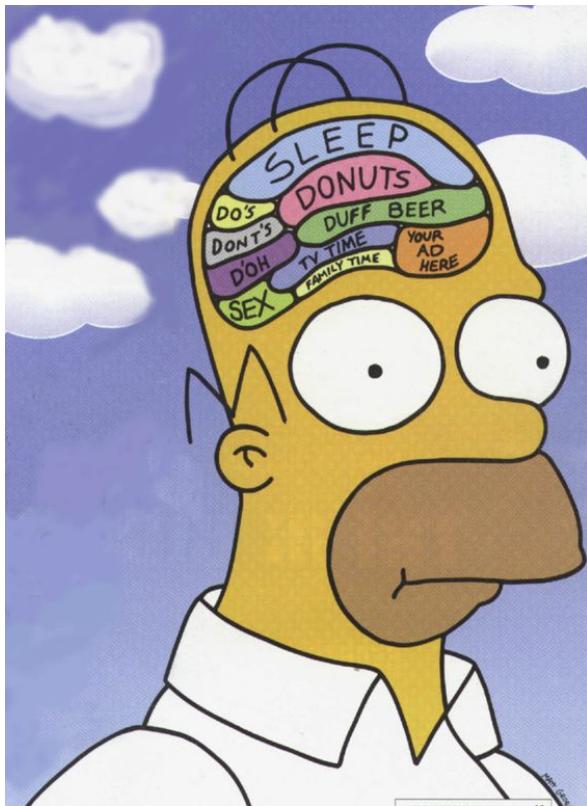
Prof. Cleyton Slaviero

Introdução

- Mundo perfeito
 - Memórias rápidas, infinitas, baratas
 - Não vivemos num mundo perfeito 😔
- Hierarquia de memória
 - Diferentes tamanhos/tipos/velocidades de memória?

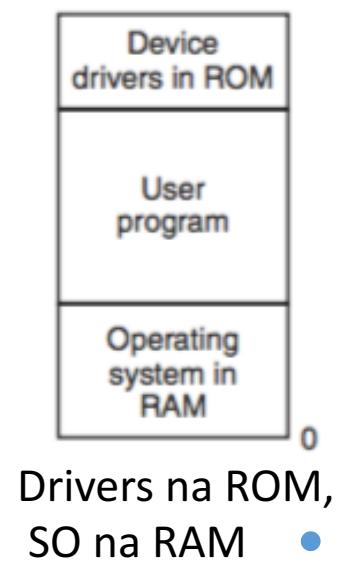
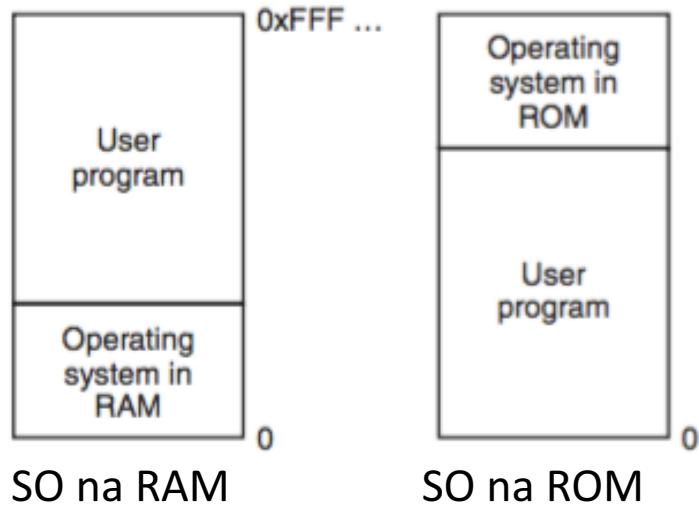


Introdução



- Como gerenciar tudo isso?
 - **Gerenciador de memória**
 - Manter o rastro de memória em uso, alocar memória para processos que precisam, e desalocar quando não precisa mais

Nenhuma abstração de memória



- O programa vê / manipula a memória diretamente
 - MOV REGISTER1,1000 #o que isso faz?
- Diferentes formas de colocar na memória
 - Como rodar diversos programas?
 - Salvando tudo em disco
 - E se dois programas tentassem acessar o mesmo endereço
 - Catástrofe! 😱
 - Paralelismo? Threads!
 - Mas pra que?

Troca de processos

- Rodando múltiplos programas
 - Troca-se de processo
 - Enquanto só um programa está na memória, tudo ok
- Com um sistema em lote, é simples e eficiente organizar a memória em partições fixas.
- Em sistemas de tempo compartilhado:
 - Pode não existir memória suficiente para conter todos os processos ativos
 - Os processos excedentes são mantidos em disco e trazidos dinamicamente para a memória a fim de serem executados



Espaços de endereçamento

- Problemas em expor a memória física
 - Programas podem destruir o sistema
 - Difícil ter vários programas rodando ao mesmo tempo
- Algum mecanismo é necessário!

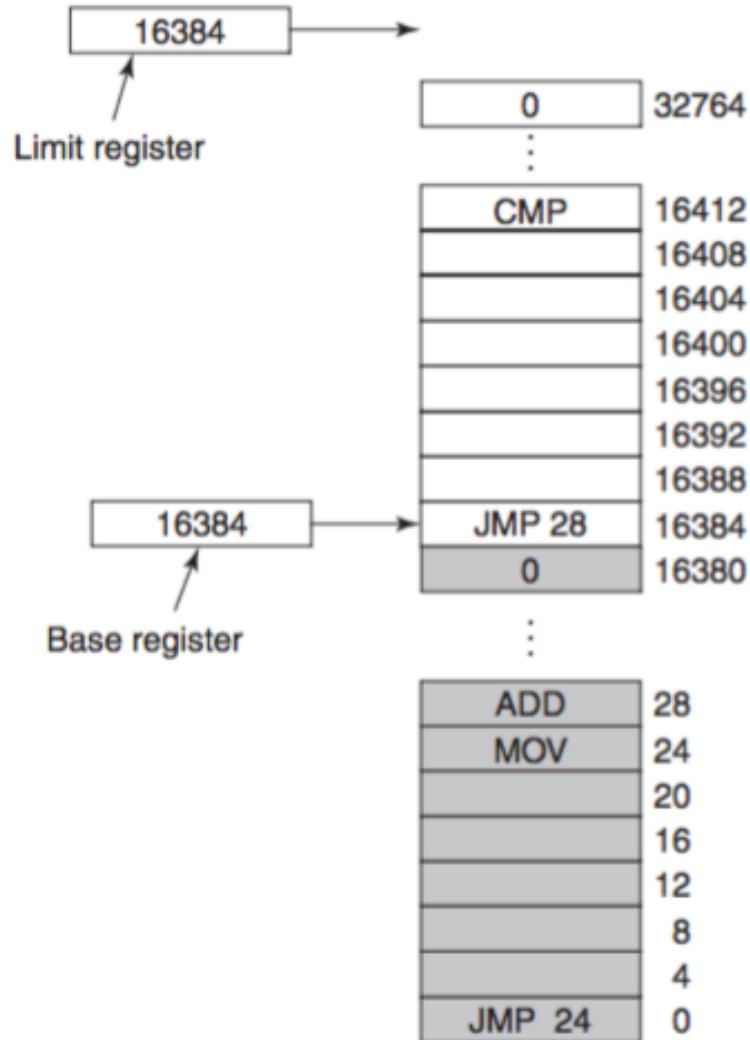
Espaços de endereçamento

- Dois problemas precisam ser resolvidos
 - Proteção
 - Realocação
- P.ex: IBM360 'taggeava' cada pedaço de memória com uma chave de proteção
 - Ineficiente
- Solução: uma nova abstração
- Espaço de endereçamento
 - Conjunto de endereços que um processo pode usar para endereçar a memória

Espaços de endereçamento

- Já usamos espaços de endereçamento
 - Endereços web (.com,.net...)
 - Números telefônicos
 - Endereços IP
- Não precisam ser numéricos

Espaços de endereçamento



- Mas, e agora... como abstrair?

- Registradores

- Alocação dinâmica
- Base e limite
- JUMP 28 → JUMP 16412
- Desvantagem: cada chamada a memória precisa de um cálculo e comparação

Troca de processos



- Se há espaço na memória para segurar todos os processos...
Ok
- Hoje em dia, isso é raro...
 - O sistema carrega processos
 - Nós carregamos processos
 - O sistema carrega mais processos
 - Seu navegador 'don't be evil' carrega mais processos...

Troca de processos



- Existem 2 abordagem gerais que podem ser usadas:
 - Swapping
 - Forma mais simples
 - Consiste em trazer totalmente cada processo para a memória, executá-lo durante um tempo e, então, devolvê-lo ao disco
 - Memória Virtual:
 - Permite que programas possam ser executados mesmo que estejam parcialmente carregados na memória principal



Universidade Federal
de Mato Grosso
Campus Rondonópolis

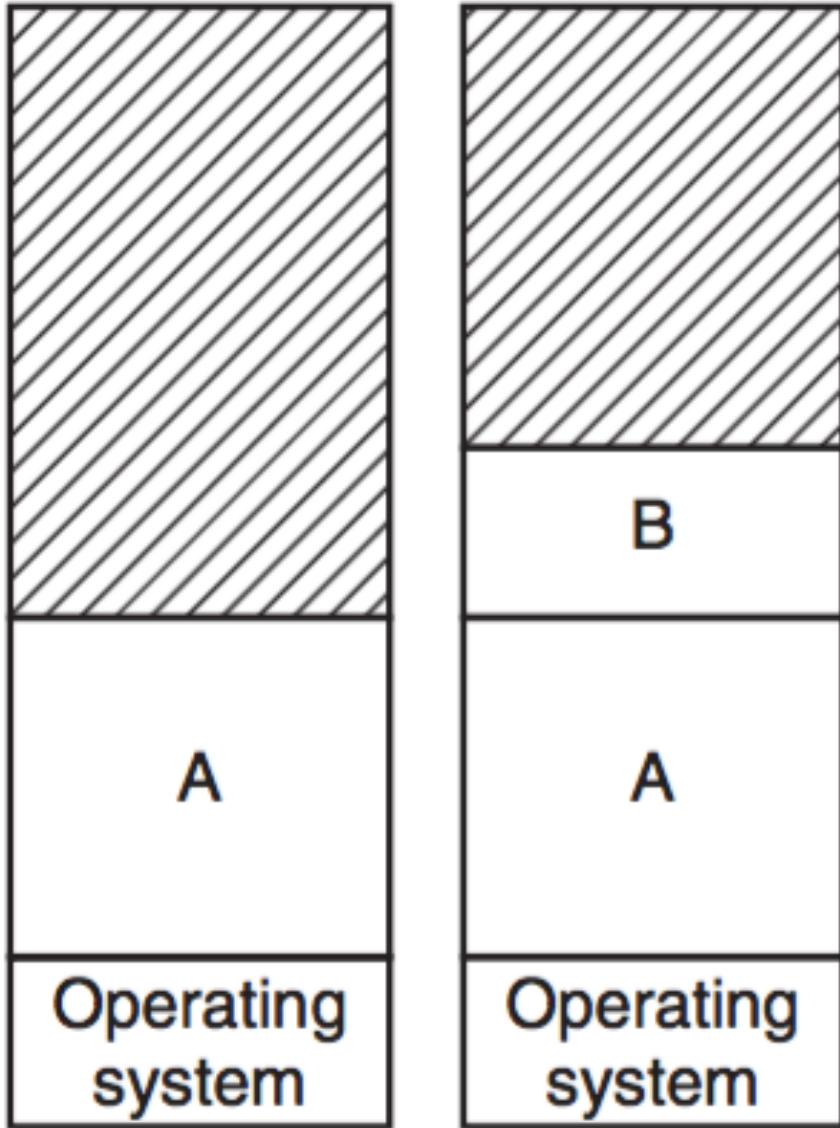
Swapping

- Chaveamento de processos inteiros entre a memória principal e o disco;
- Swap-out
 - Da memória para uma região especial do disco (swap)
- Swap-in
 - Do disco para a memória
- Pode ser utilizado tanto com partições fixas quanto com partições variáveis



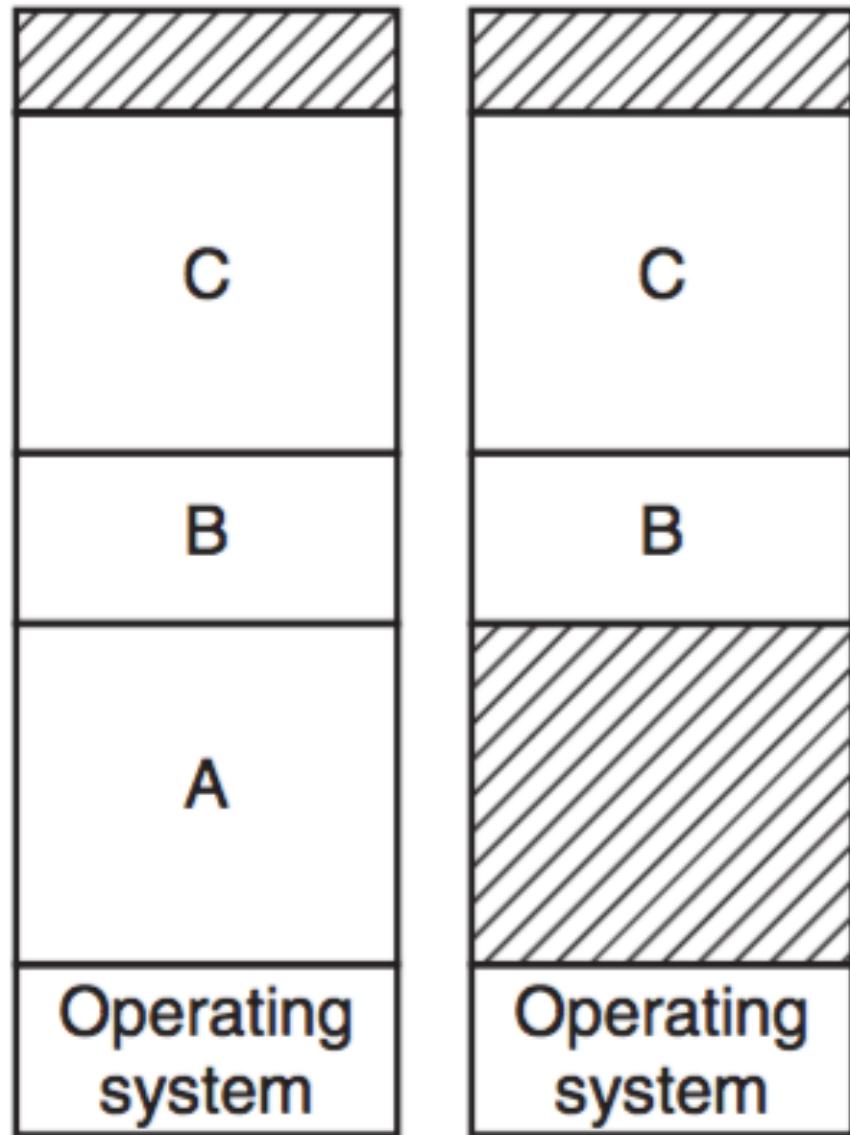
Swapping

- Operação
 - Inicialmente, somente o processo A está na memória
 - Então B ou é criado ou trazido de volta (swap in) à memória



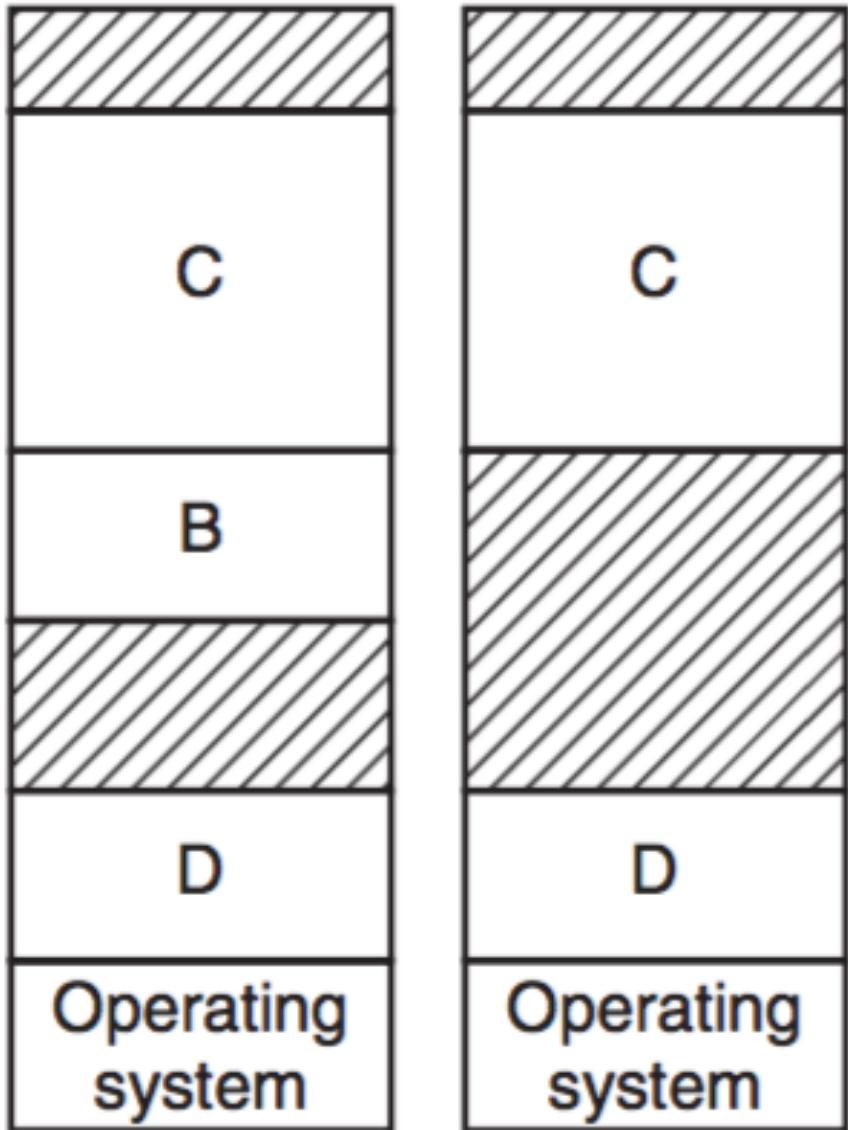
Swapping

- Operação
 - Em seguida, C é criado ou trazido de volta (swap in) à memória
 - A então é levado ao disco (swap out)



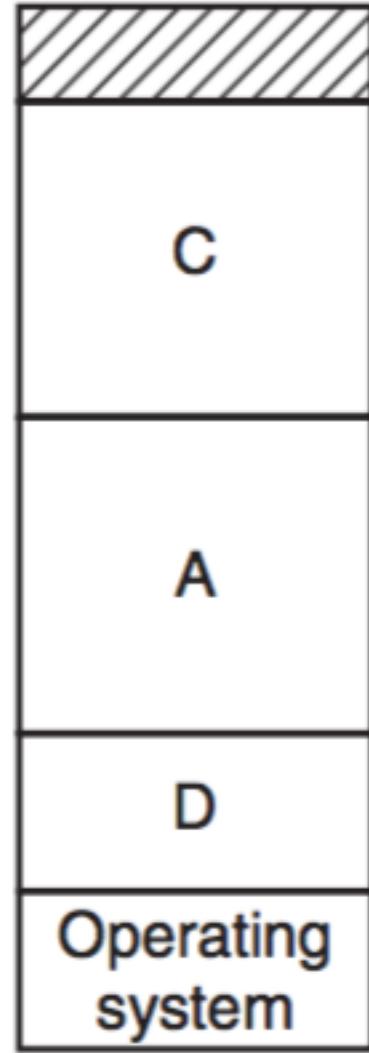
Swapping

- Operação
 - E D é trazido à memória (ou iniciado)
 - B então sai

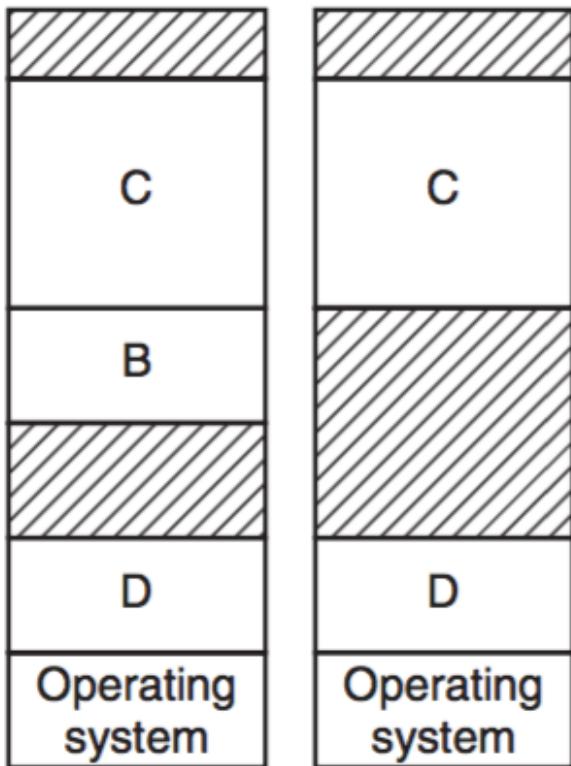


Swapping

- Operação
 - E A é trazido de volta
- Note que
 - A agora está em outra porção da memória → devemos realocar os endereços
 - As partições são de tamanho variável



Swapping



- Problema

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

- Solução: ???

Swapping

- Problema
 - A operação de swap pode criar muitos “buracos” na memória
 - Problema de fragmentação externa
- Solução: **Compactação de memória**
 - Mova todos os processos o mais para baixo possível na memória – haverá um único espaço vazio acima
 - Consome bastante tempo de CPU

Swapping – Alocando memória

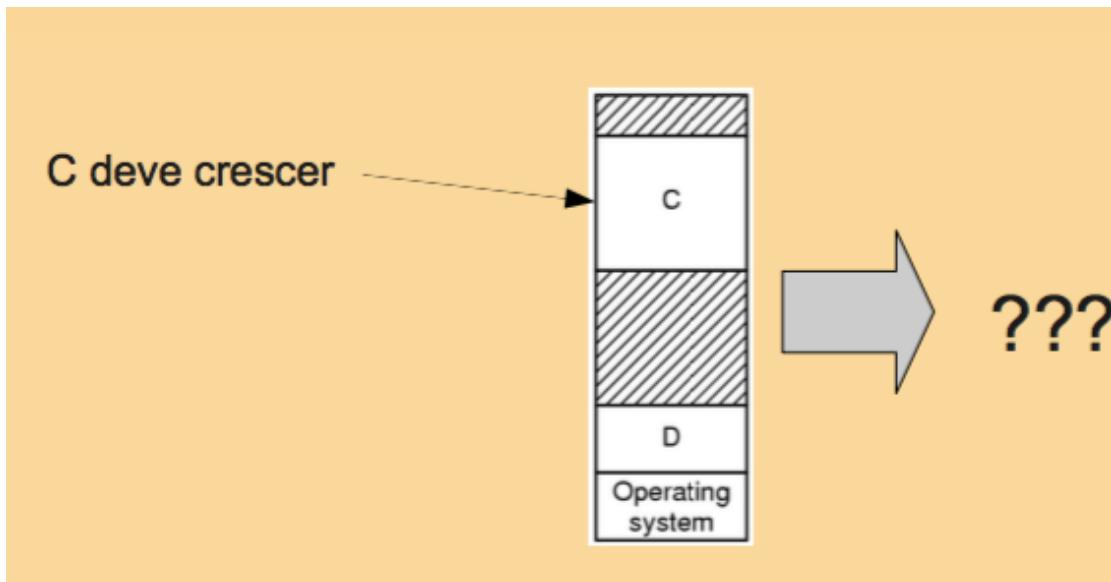
- Mais problemas...
 - A alocação de memória muda à medida em que
 - Os processos chegam à memória
 - Os processos deixam a memória
 - Quanto de memória devemos alocar a um processo quando ele é criado ou trazido a ela?

Swapping – Alocando memória

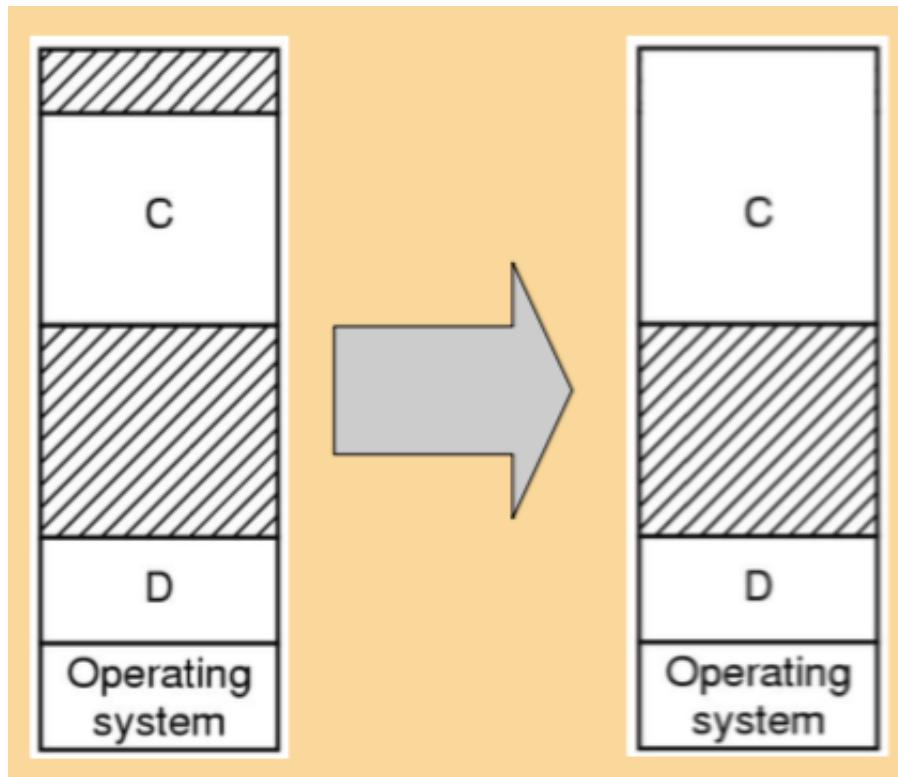
- Mais problemas...
 - A alocação de memória muda à medida em que
 - Os processos chegam à memória
 - Os processos deixam a memória
 - Quanto de memória devemos alocar a um processo quando ele é criado ou trazido a ela?
 - Se eles tiverem tamanho fixo, aloque o que ele precisa

Swapping – Alocando memória

- Mas e se o segmento de dados deles cresce com o tempo?



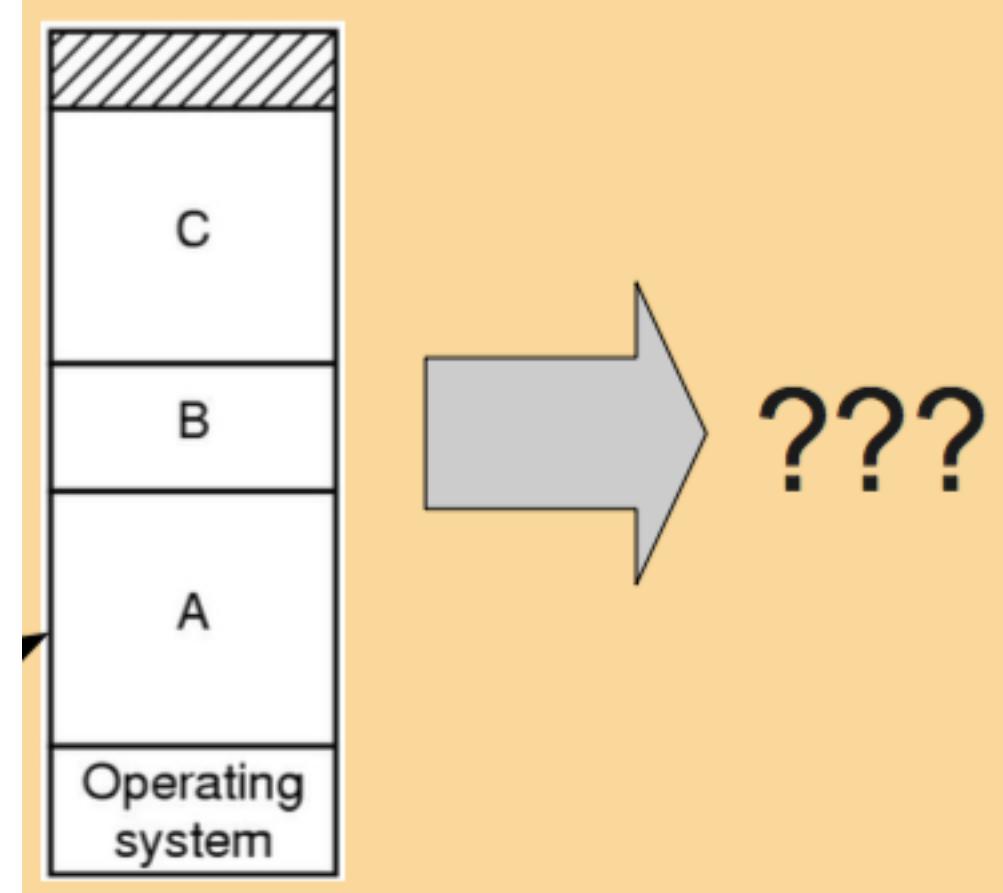
Swapping – Alocando memória



- Mas e se o segmento de dados deles cresce com o tempo?
 - Se houver um “buraco” adjacente à memória atual do processo, ele pode ser alocado, e o processo cresce nesse buraco

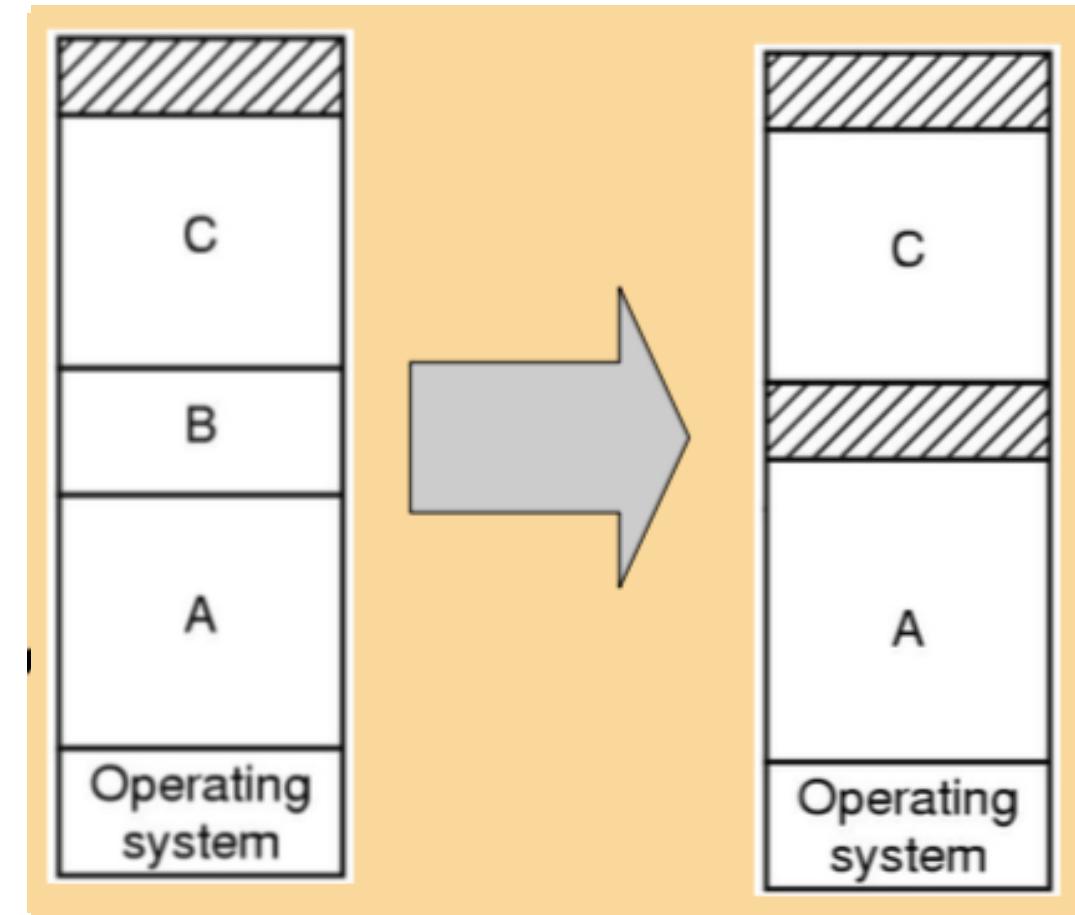
Swapping – Alocando memória

- E se ele for adjacente a outro processo?



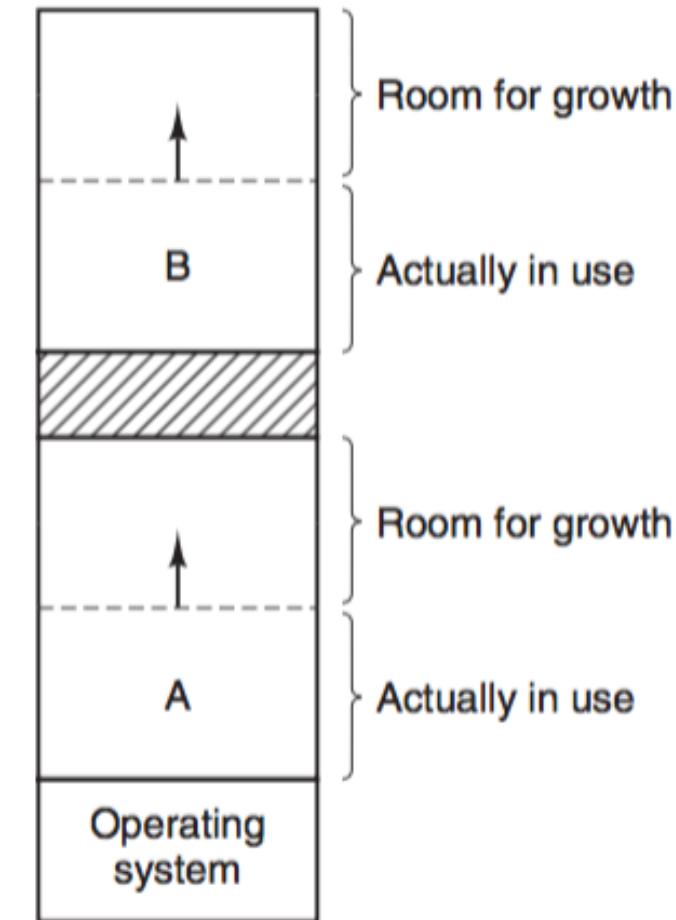
Swapping – Alocando memória

- 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 removidos para criar esse buraco



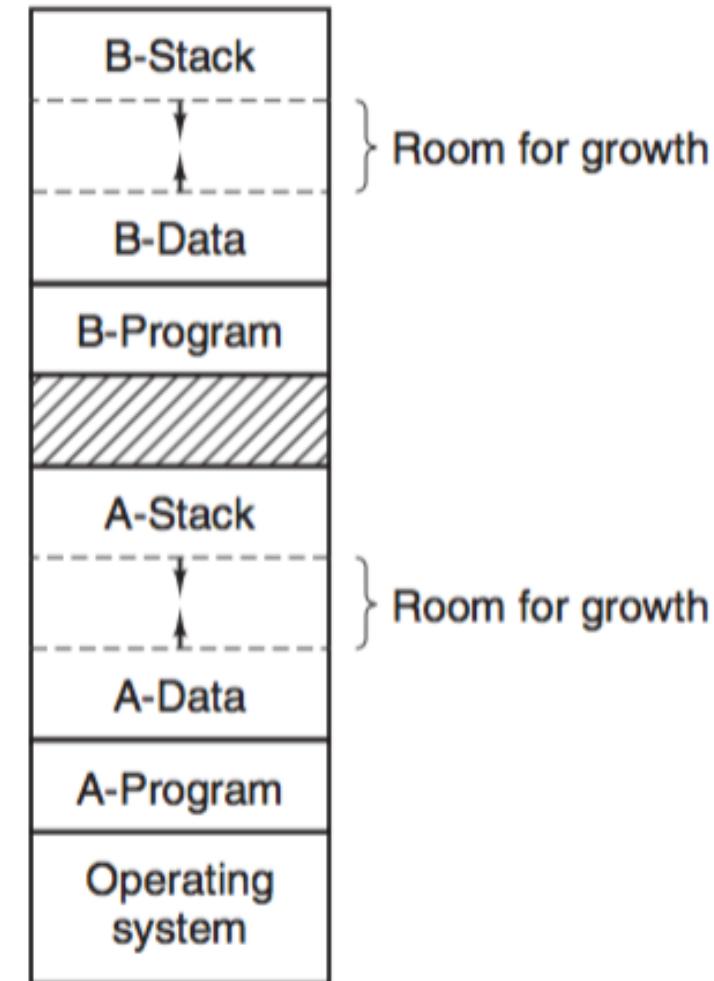
Swapping – Alocando memória

- Alternativamente, podemos alocar memória a mais para cada processo carregado
 - Reduz o overhead de ter que fazer swap
- Se ainda assim o processo for para o disco, somente a memória realmente em uso é gravada



Swapping – Alocando 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
- Se ainda assim ficar sem espaço
 - Swap, como antes
 - Não há espaço no swap?
 - Processo é suspenso



Swapping

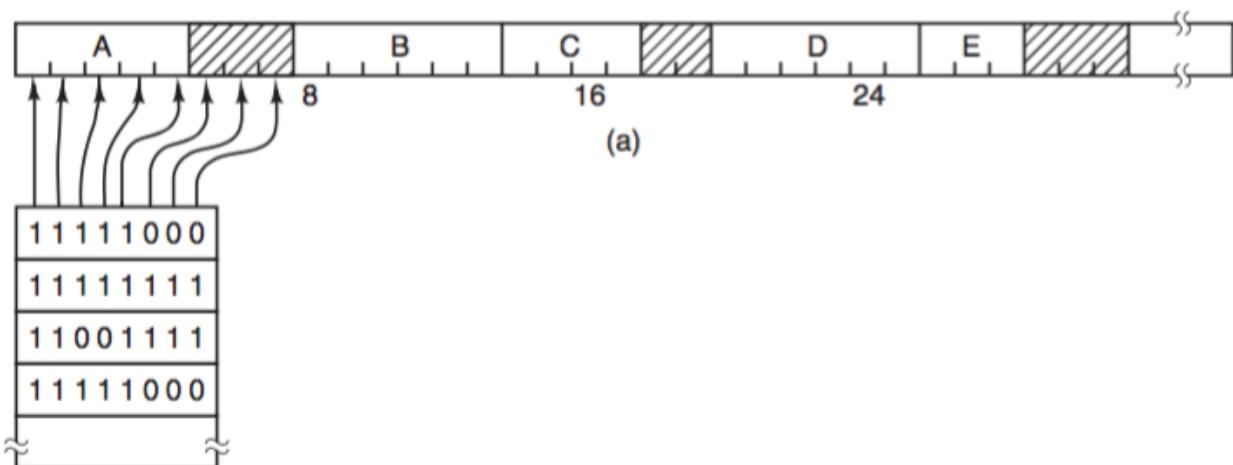
Gerenciamento do uso da memória



- Mapas de Bits (Bitmaps):
 - Memória é dividida em **unidades de alocação**
 - Pode conter vários KB
 - Cada unidade corresponde a um bit no bitmap:
 - 0 → livre
 - 1 → ocupado
 - Tamanho do bitmap depende do tamanho da unidade e do tamanho da memória;
 - unidades de alocação pequenas → bitmap grande;
 - unidades de alocação grandes → perda de espaço;

Swapping

Gerenciamento do uso da memória



- Mapas de Bits (Bitmaps):

- Memória é dividida em **unidades de alocação**
 - Pode conter vários KB
- Cada unidade corresponde a um bit no bitmap:
 - 0 → livre
 - 1 → ocupado

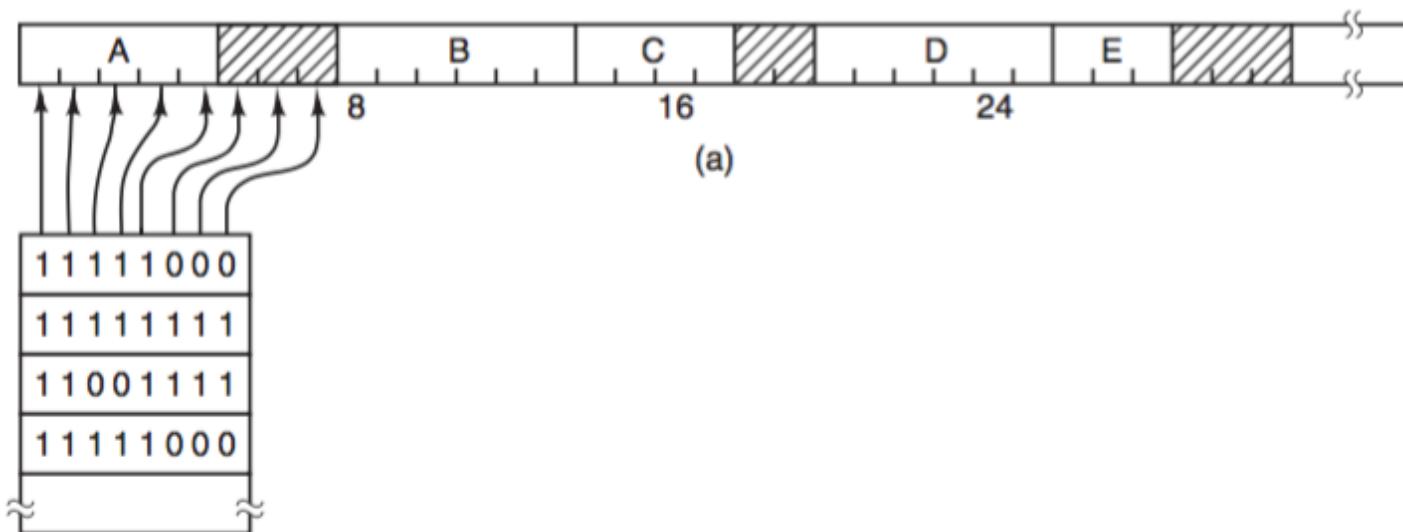
Tamanho do bitmap depende do tamanho da unidade e do tamanho da memória;

- unidades de alocação pequenas → bitmap grande;
- unidades de alocação grandes → perda de espaço;

Swapping

Técnicas para alocação dinâmica de memória

- Mapas de Bits (Bitmaps):
 - Problema
 - Quando um novo processo (de k unidades) é trazido à memória, o gerenciador deve buscar, no bitmap, uma seqüência de k zeros consecutivos – operação potencialmente lenta

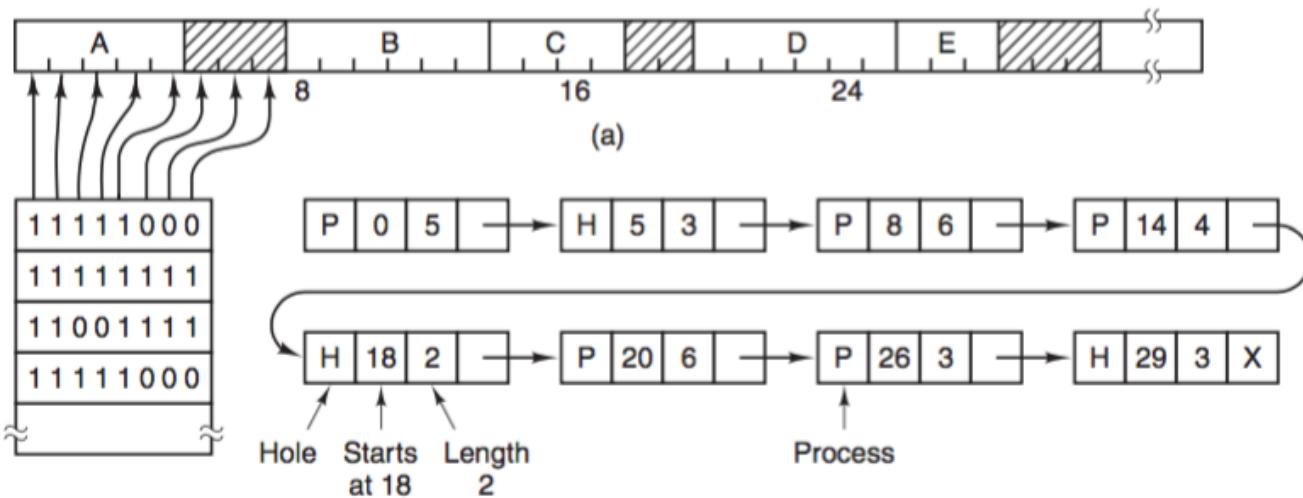


Swapping

Técnicas para alocação dinâmica de memória

- Lista Ligada

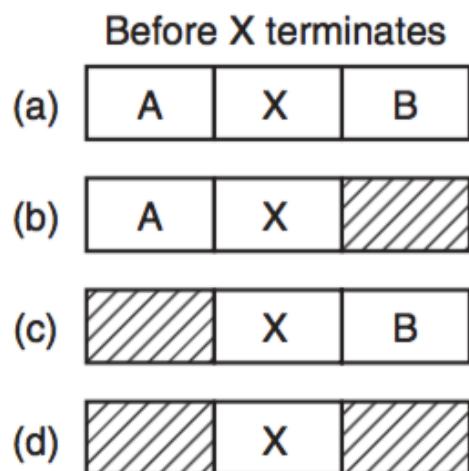
- Manter uma lista ligada de segmentos de memória livres e alocados
- Cada segmento ou contém um processo ou é um buraco vazio entre dois processos
 - Ambos tem endereço de início, tamanho, ponteiro para o próximo item



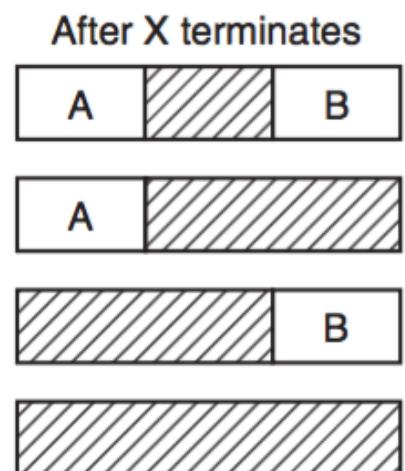
Swapping

Técnicas para alocação dinâmica de memória

- Listas (duplamente) Ligadas
 - A lista ordenada por endereço fica fácil de ser atualizada



becomes
becomes
becomes
becomes



Troque P por H na posição correspondente da lista

Elimine o nó que representava X, atualizando o seguinte e o anterior



Swapping

Técnicas para alocação dinâmica de memória

- Existem três métodos que podem ser usados para selecionar uma região para um processo.
 - Se a lista estiver ordenada por endereço

Swapping

Técnicas para alocação dinâmica de memória

- Os algoritmos de alocação são:
 - Primeira escolha (First fit):
 - Percorre a lista de segmentos até que encontre um buraco grande o bastante
 - Quebre o buraco em dois pedaços – um para o processo e um para a memória não usada
 - Variação: inicie a busca a partir de onde parou, na vez anterior (next fit)

Swapping

Técnicas para alocação dinâmica de memória

- Os algoritmos de alocação são:
 - Melhor escolha (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

Swapping

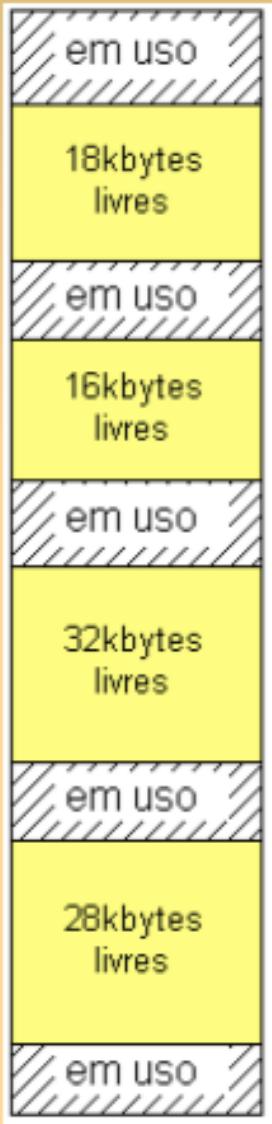
Técnicas para alocação dinâmica de memória

- Os algoritmos de alocação são:
 - Pior escolha (worst fit):
 - Sempre tome o maior buraco disponível
 - Reduz a chance de buracos minúsculos e inúteis
 - No geral, todos tem algum problema

Dois processos, P e Q, são alocados nessa ordem:

14kbytes (P)

20kbytes (Q)



Sentido em que a memória é percorrida

Áreas livres iniciais

Melhor Escolha

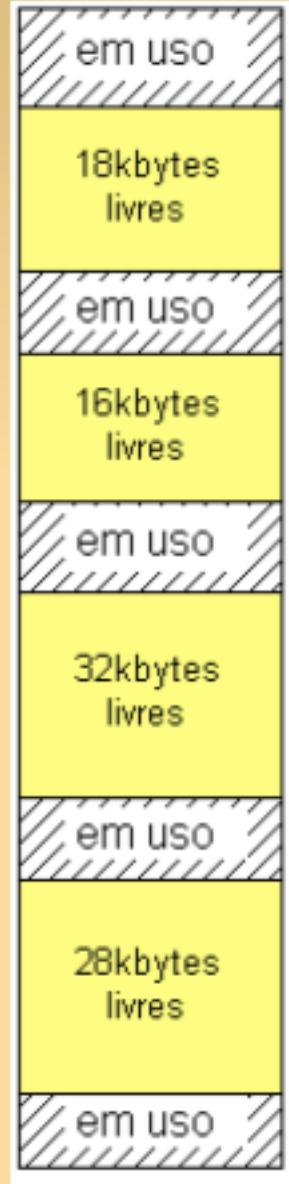
Pior Escolha

Primeira Escolha

Dois processos, P e Q, são alocados nessa ordem:

14kbytes (P)
20kbytes (Q)

Sentido em que a memória é percorrida



Áreas livres iniciais



Melhor Escolha

Pior Escolha

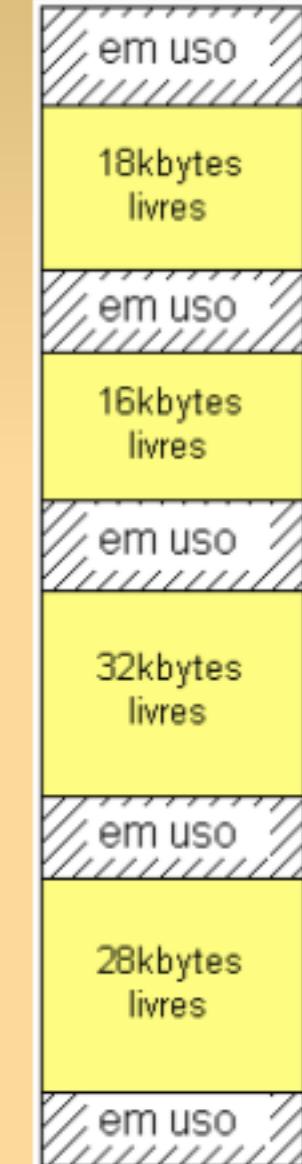
Primeira Escolha

Dois processos, P e Q, são alocados nessa ordem:

14kbytes (P)

20kbytes (Q)

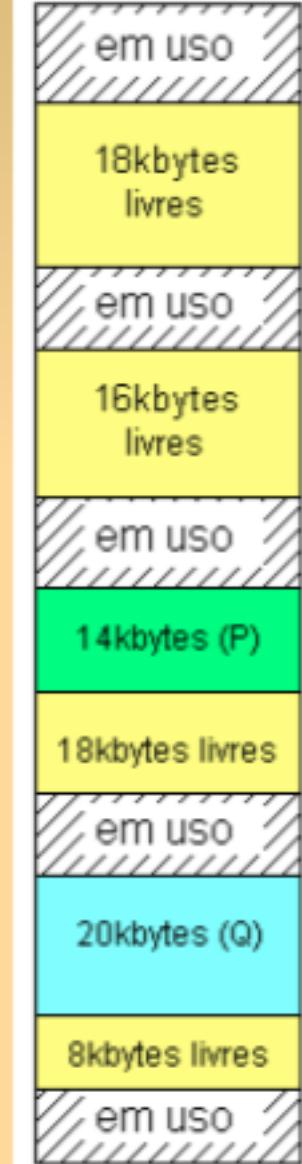
Sentido em que a memória é percorrida



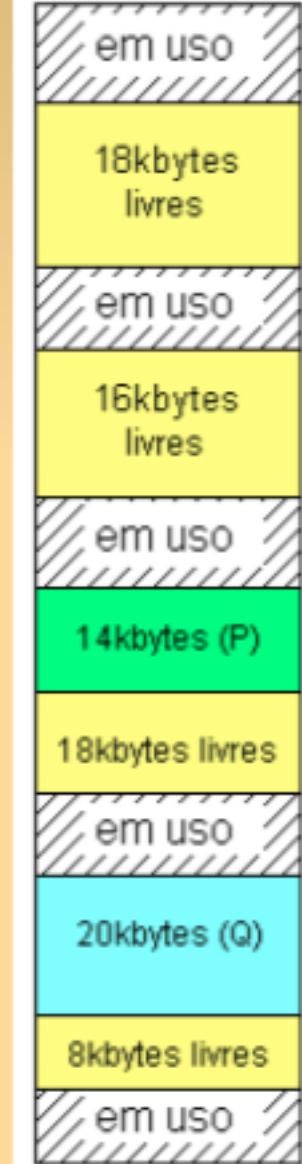
Áreas livres iniciais



Melhor Escolha



Pior Escolha

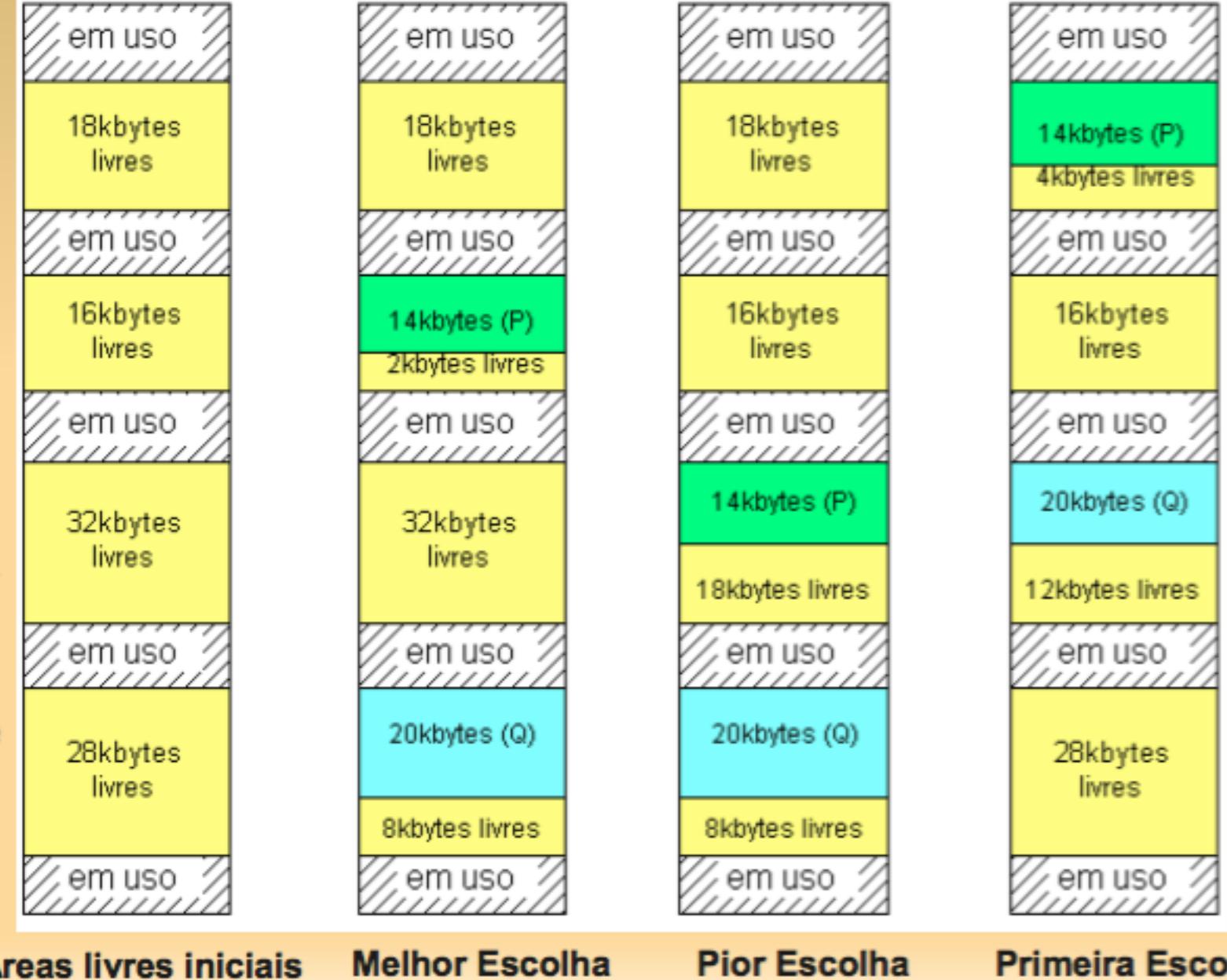


Primeira Escolha

Dois processos, P e Q, são alocados nessa ordem:

14kbytes (P)
20kbytes (Q)

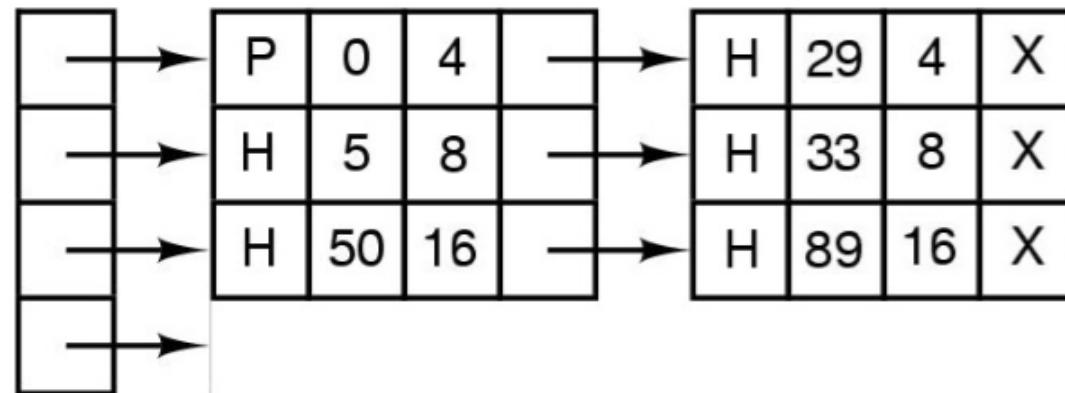
Sentido em que a memória é percorrida



Swapping

Técnicas para alocação dinâmica de memória

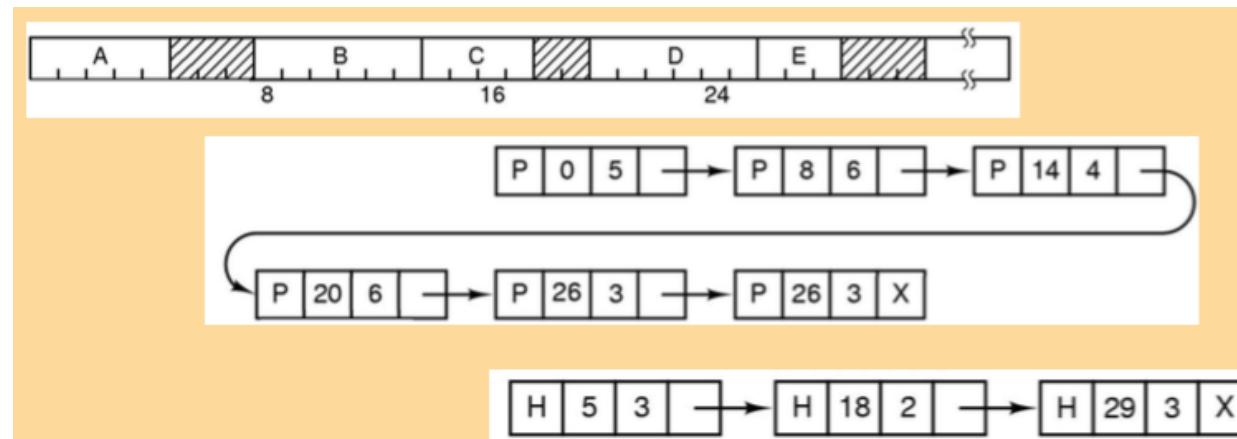
- Os algoritmos de alocação são:
 - Quick fit
 - Mantém listas separadas para alguns dos tamanhos mais comumente requisitados
 - Uma para cada tamanho
 - Problema quando processo é desalocado:
 - Encontrar os vizinhos ao buraco que ele deixou, para união



Swapping

Técnicas para alocação dinâmica de memória

- Algoritmos de alocação
 - Todos podem ser melhorados se mantivermos buracos e processos em listas separadas
 - Há ganho de desempenho
 - Ainda assim, quando um processo é retirado da memória, mesclar buracos é de alta complexidade.



Swapping

Técnicas para alocação dinâmica de memória

- Principais consequências dos algoritmos:
 - Primeira escolha: 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.
 - Melhor escolha: deixa o menor resto, porém após um longo processamento poderá deixar “buracos” muito pequenos para serem úteis.
 - Pior escolha: 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 jobs.

Memória virtual

- O tamanho das memórias cresce rapidamente
 - Problema: o tamanho do software cresce muito mais rápido
 - Há necessidade de rodar programas grandes demais para a memória
- Mas e o swapping?
 - Problema: velocidade

Memória virtual

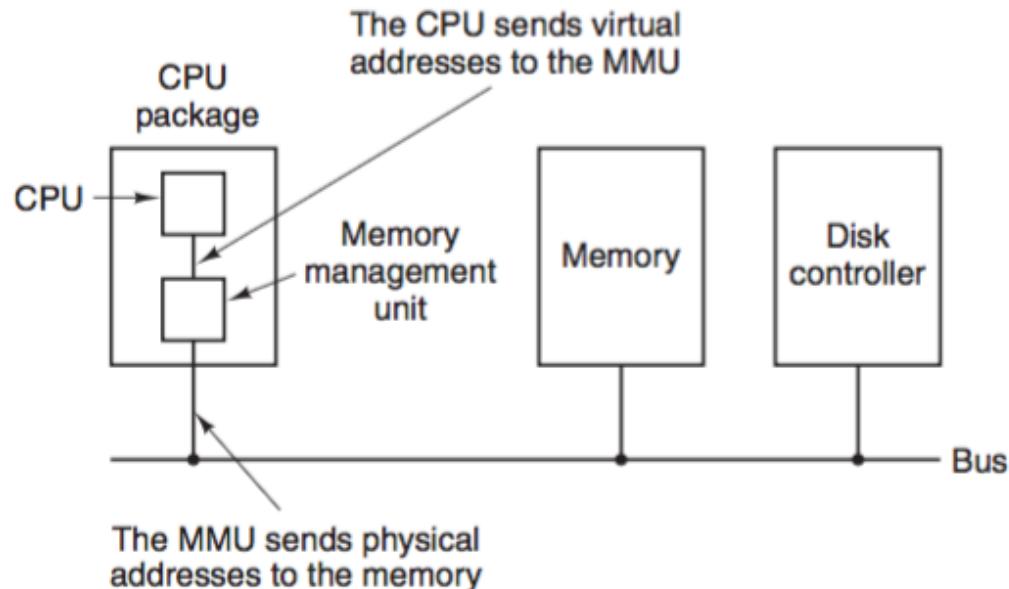
- Uma solução ('60s): overlays
 - Separar o programa em partes
 - O gerenciador de overlays era carregado na memória
 - Cada pedaço era carregado de acordo
 - Problema: gerenciar tudo isso
- Porque não deixar o computador faze isso?
 - **Memória virtual**

Memória virtual

Paginação

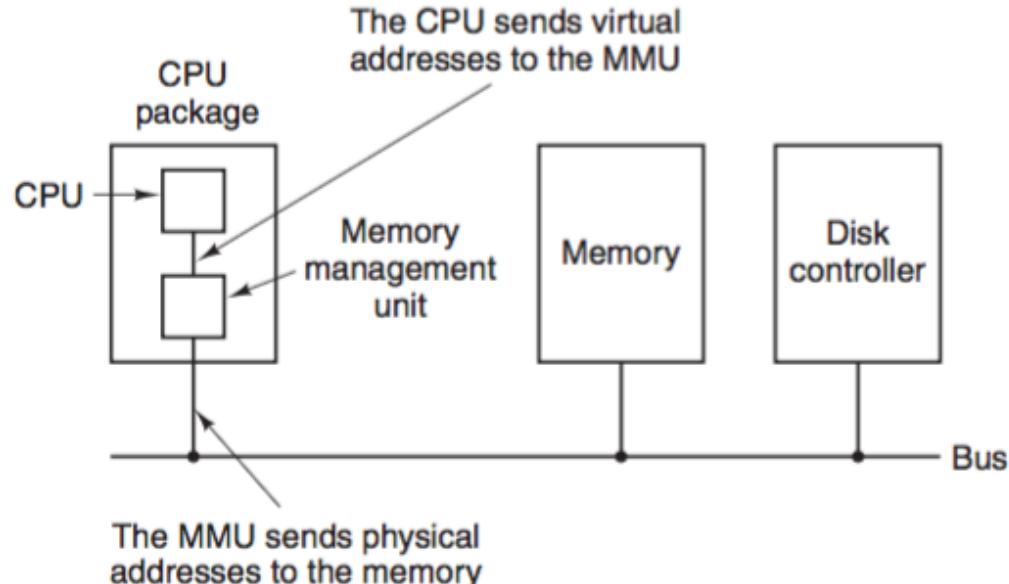
- Intuição
 - Cada programa tem seu espaço de endereçamento quebrado em páginas
 - Cada página é um espaço contíguo de endereços, mapeados a memória física
 - Nem todas as páginas tem que estar na memória física ao mesmo tempo
 - O hardware faz o mapeamento 'on the fly' quando necessário

Memória virtual Paginação



- Programas fazem referência à memória
 - Ex: MOV REG,1000
 - Não necessariamente esse é o endereço real
- Os endereços físicos (reais) podem ser gerados de muitas maneiras:
 - Mapeamento direto
 - Usando registradores base ...
 - ...etc
- Endereços referenciados pelos programas são chamados de virtuais
 - Na ausência de memória virtual, o endereço virtual bate com o físico
 - Com memória virtual, é tratado pela MMU (Memory Management Unit)

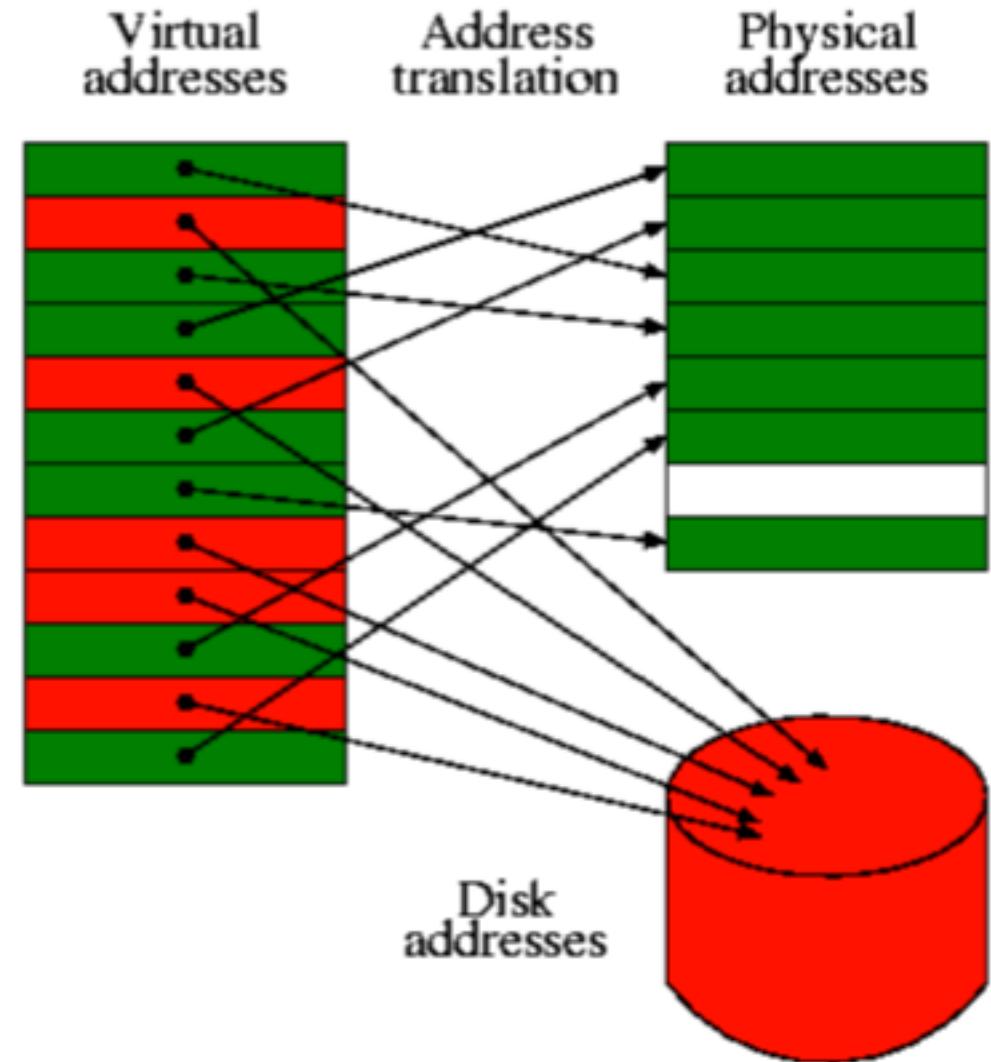
Memória virtual Paginação



- Espaço de Endereçamento Físico de um processo:
 - Formado por todos os endereços físicos/reais aceitos pela memória principal (RAM);
- Espaço de Endereçamento Virtual de um processo:
 - Formado por todos os endereços virtuais que esse processo pode gerar;

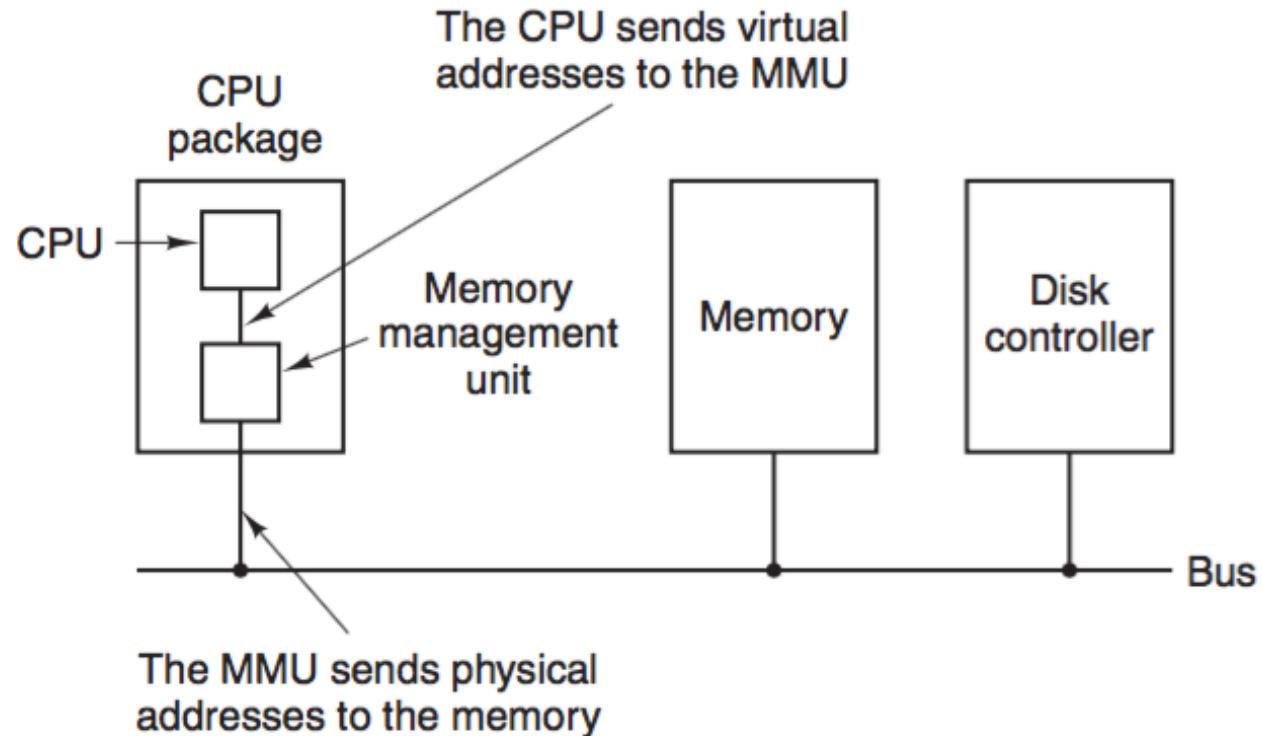
Memória virtual

- Um processo em Memória Virtual faz referência a endereços virtuais e não a endereço reais de memória RAM;
- No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço real, pois a CPU manipula apenas endereços reais da memória RAM → MAPEAMENTO;



Memória virtual

- A MMU mapeia o endereço virtual à memória física

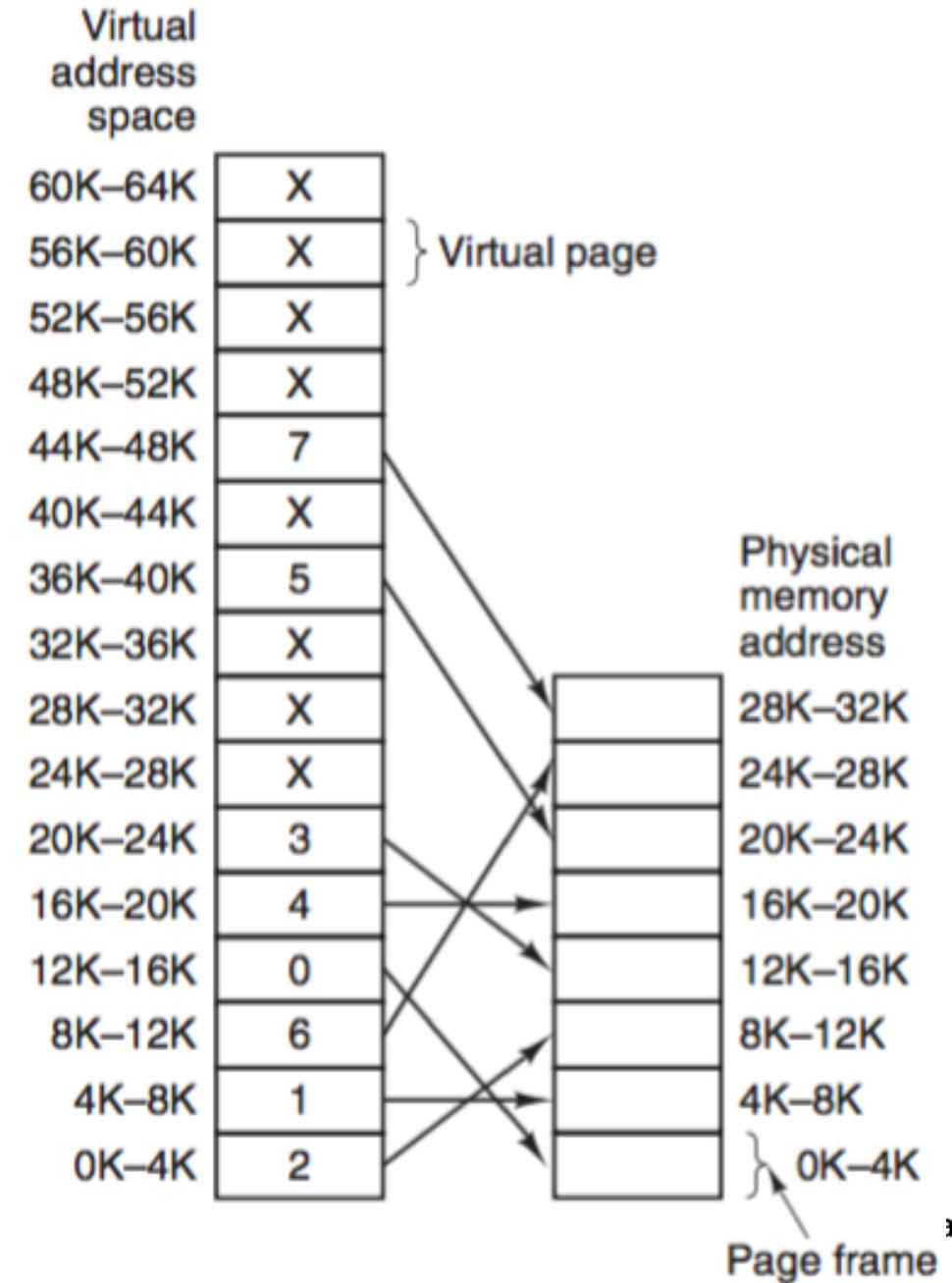


Memória virtual

- Páginas
 - Quando o programa referencia uma parte de seu endereço que está na memória, o hardware faz o mapeamento na hora
 - Se, contudo, essa parte não estiver na memória:
 - O S.O. traz do disco o pedaço que falta
 - Executa novamente a instrução que falhou
 - Enquanto isso, o escalonador pode colocar outro processo para rodar
- Memória virtual é semelhante ao swap
 - Swap aloca espaço para todo o processo, MV não

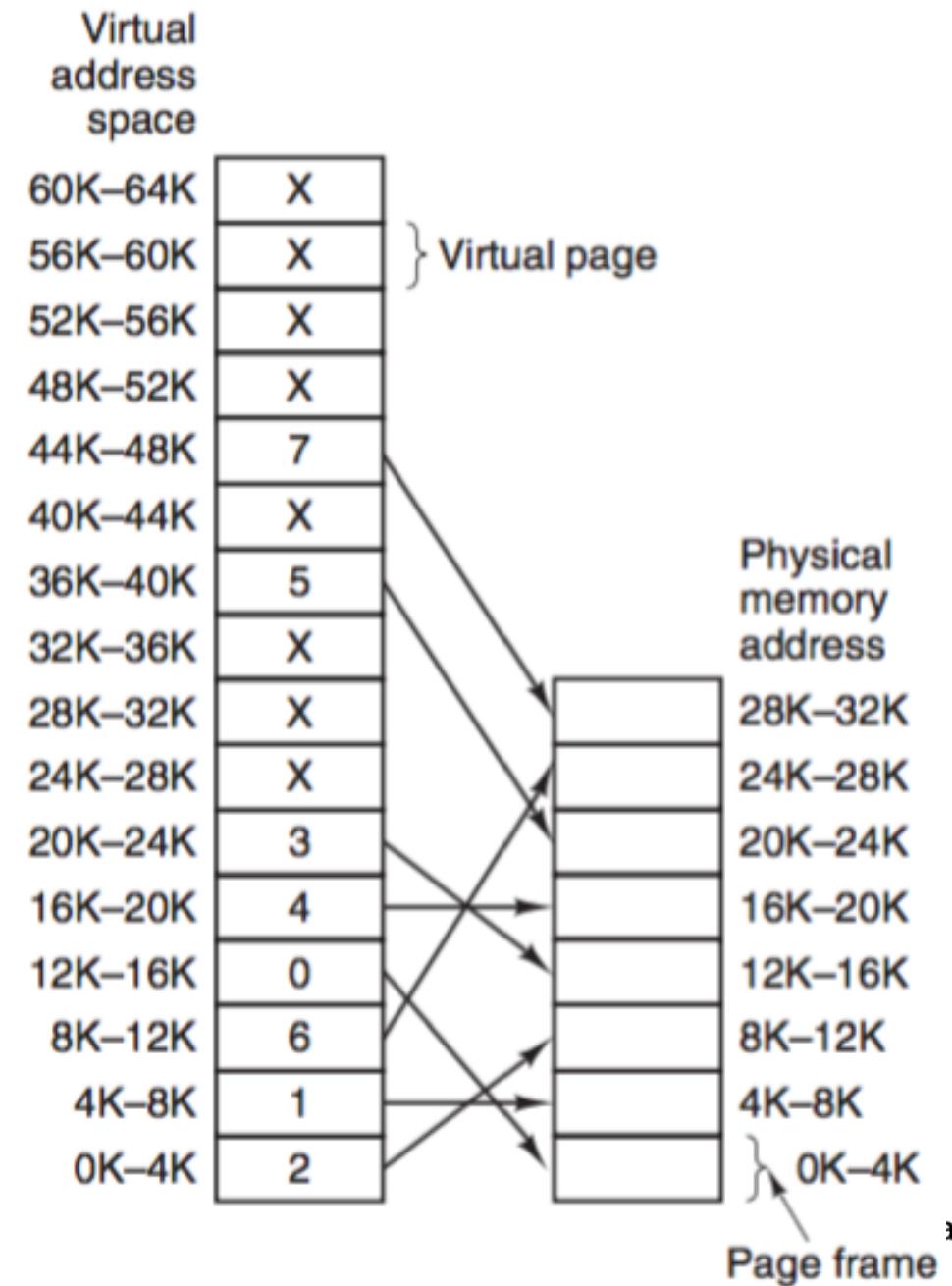
Memória virtual

- Exemplo
 - Computador que gera 64k de endereços virtuais
 - Tem apenas 32K físicos
 - Programas não conseguem ser carregados inteiramente na memória



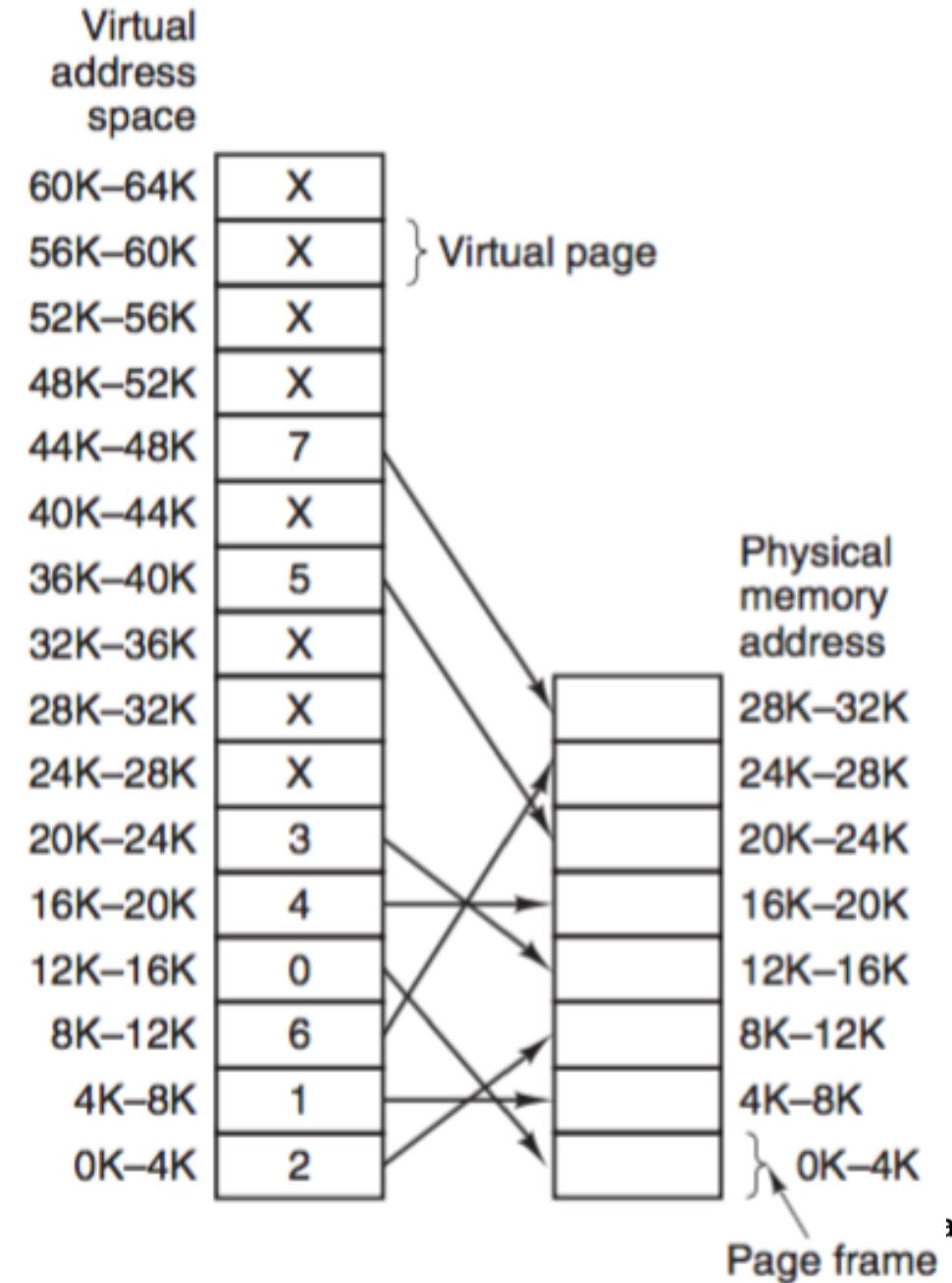
Memória virtual

- Divide-se o espaço de endereçamento virtual em unidades de tamanho fixo (páginas)
 - Aqui, páginas de 4K
 - Unidades correspondentes na memória física são os page frames
 - Geralmente páginas e page frames tem o mesmo tamanho



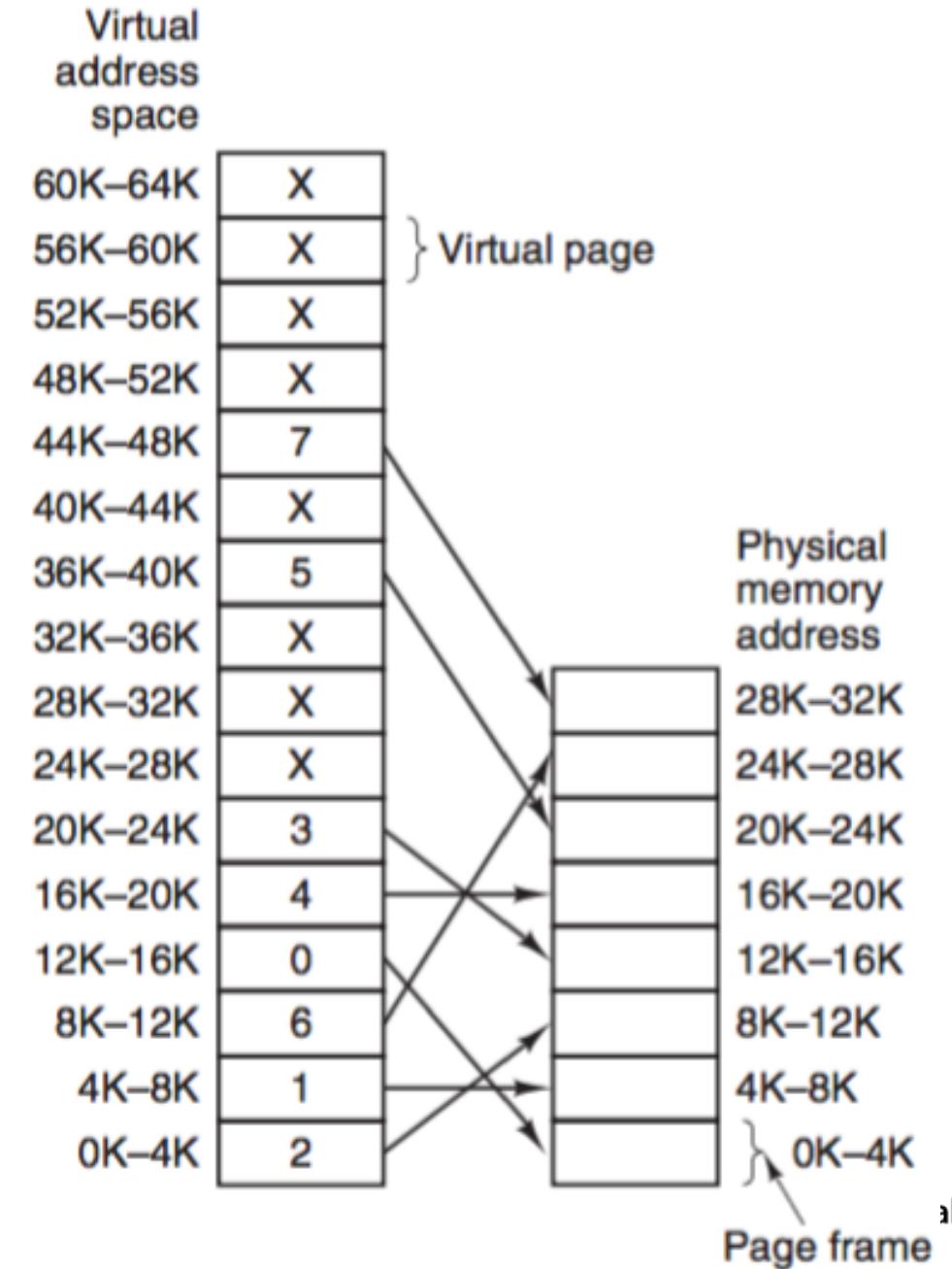
Memória virtual

- Embora tenha 32KB, o sistema age como se tivesse 64KB
- **MOV REG, 5**
 - A MMU identifica que é a primeira página (5B acima da sua base $\rightarrow 0$)
 - Ela está mapeada à terceira frame, que começa em 8k = 8192
 - O endereço enviado ao barramento é $5 + 8192 = 8197$



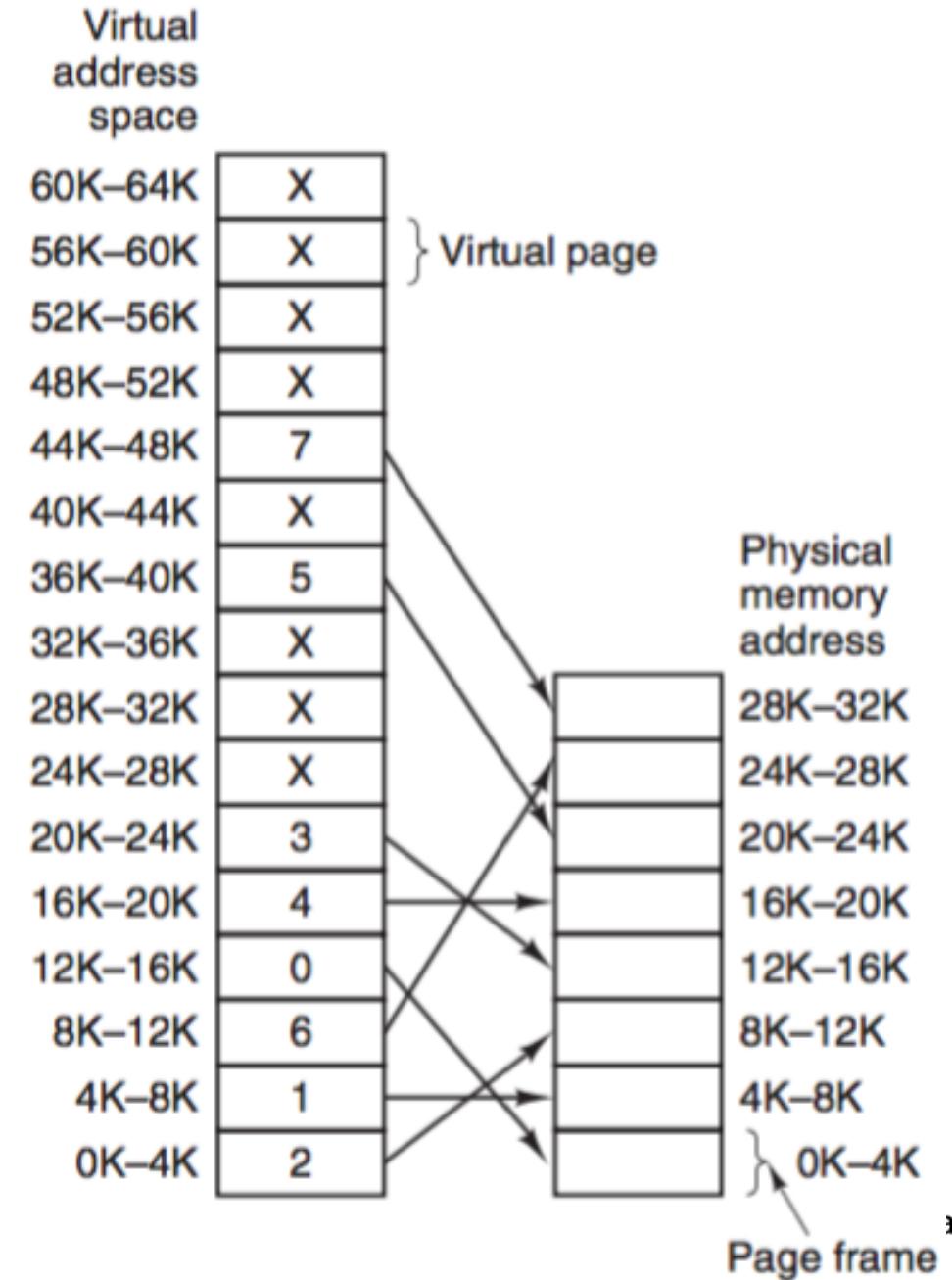
Memória virtual

- Como sabemos que as páginas estão na memória efetivamente?



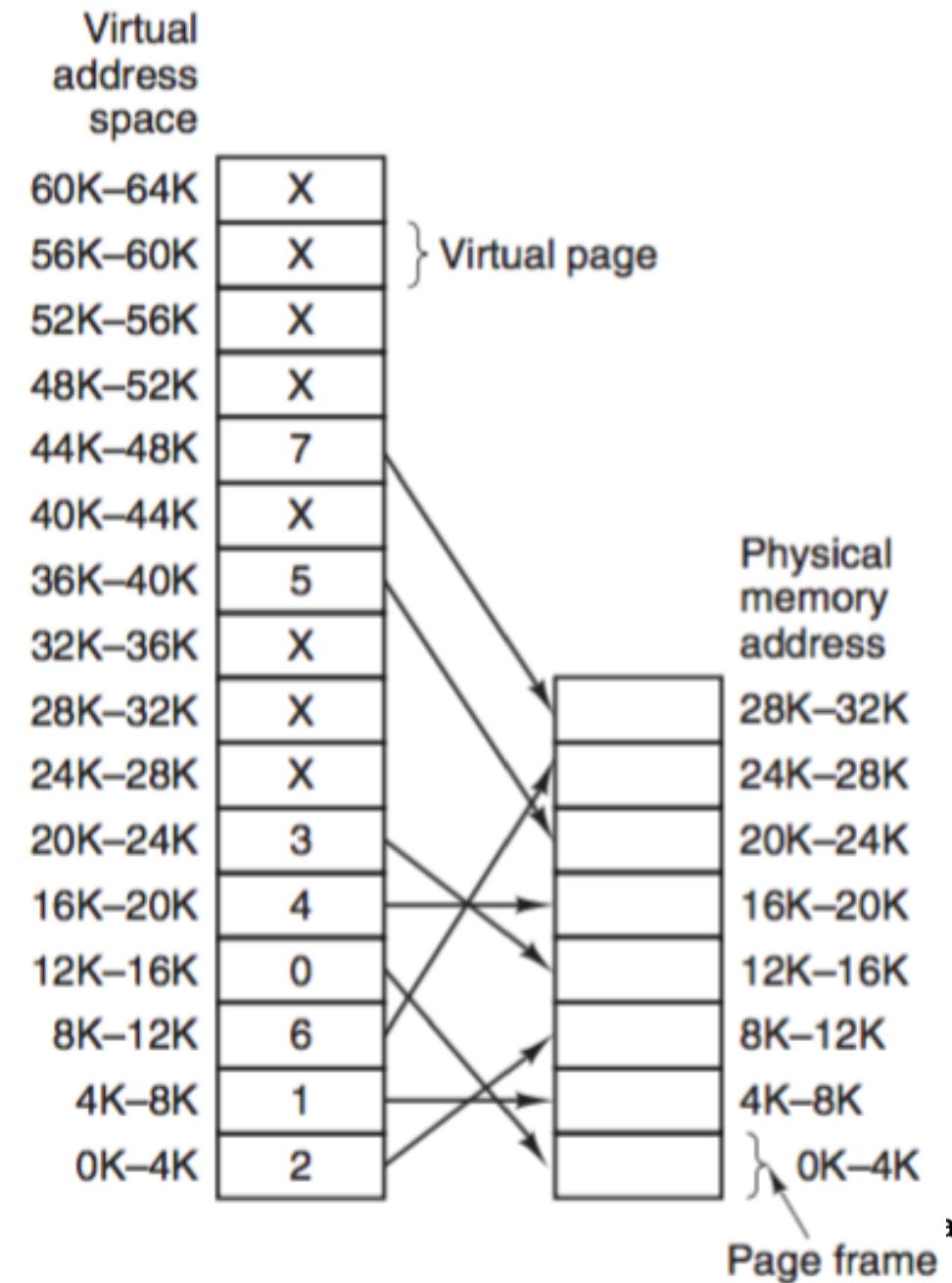
Memória virtual

- Como sabemos que as páginas estão na memória efetivamente?
 - Se temos apenas 8 frames, somente 8 páginas (das 16) estão mapeadas
 - Solução:
 - Bit de presente/ausente
 - Identifica que páginas estão fisicamente presentes na memória



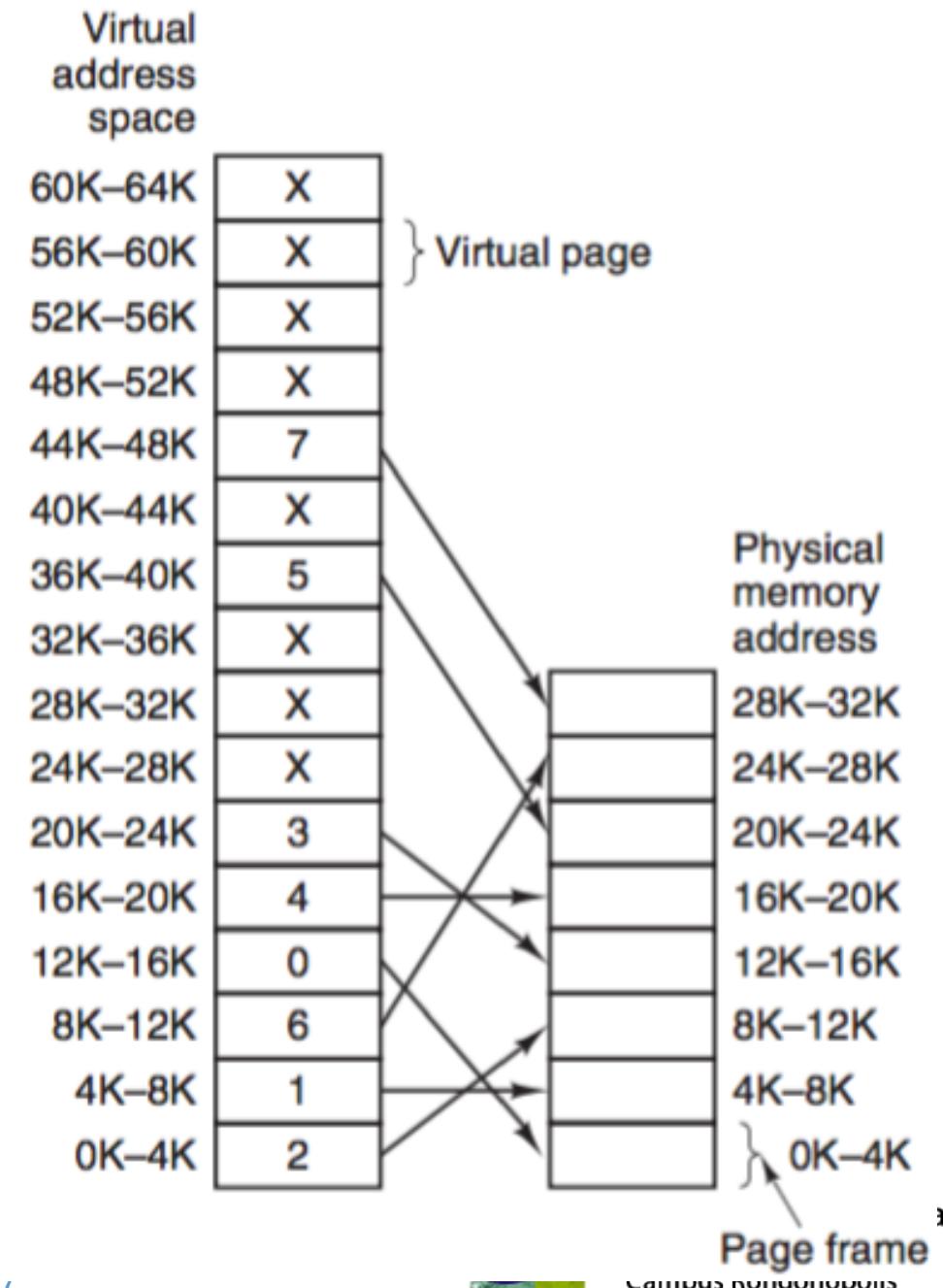
Memória virtual

- E se um programa referenciar um endereço não mapeado?
 - A MMU verifica que a página não está mapeada
 - Força o desvio da CPU para o SO, via interrupção
 - PAGE FAULT



Memória virtual

- E se um programa referenciar um endereço não mapeado?
 - O S.O. toma um page frame pouco usado
 - Escreve seu conteúdo no disco
- Carrega a página recém referenciada na frame recém liberada
 - Muda o mapa (veremos mais adiante)
 - Reinicia a instrução aprisionada no trap (que causou a interrupção)



Memória virtual

- Então...
 - Memória Virtual pode ser implementada quebrando-se o espaço de endereçamento virtual em páginas
 - E mapeando cada página a alguma moldura de página na memória física
 - Ou mantendo-a temporariamente não mapeada

Memória virtual

- Técnicas de MV:
 - Paginação:
 - Blocos de tamanho fixo (endereços contínuos) chamados de páginas;
 - SO mantém uma fila de todas as páginas;
 - Endereços Virtuais formam o espaço de endereçamento virtual;
 - O espaço de endereçamento virtual é dividido em páginas virtuais;
 - Mapeamento entre endereços reais e virtuais (MMU);

Memória virtual

- Técnicas de MV
 - Segmentação:
 - Blocos de tamanho arbitrário chamados segmentos;
 - Arquitetura (hardware) tem que possibilitar a implementação tanto da paginação quanto da segmentação;