

Aula 8

Professores:

Felipe M. G. França
Valmir C. Barbosa

Conteúdo:

Gerenciamento de memória

- Introdução
- Principais métodos de gerenciamento

Introdução

→ A maioria dos computadores possui uma hierarquia de memória:



A memória mais acima na hierarquia de memória (registradores), são as mais caras e rápidas, e por isso, são as menores, além de serem voláteis. As memórias secundárias, como o disco e o CDROM, são as mais baratas, e com isso, são as maiores, além de serem não-voláteis.

→ O gerenciamento da hierarquia de memória pelo **gerenciador de memória** é importante:

- Devemos saber quais partes da memória estão alocadas aos processos, e quais partes estão disponíveis.
- Devemos gerenciar a alocação e a desalocação da memória dinamicamente.
- Deveremos gerenciar a troca entre a memória e o disco, caso a memória seja insuficiente para armazenar todos os processos.

Introdução

- Os sistemas de gerenciamento de memória podem ser divididos em duas classes:
 - Os que mantém todo o processo na memória durante toda a sua execução:
 - Monoprogramação sem troca ou paginação
 - Multiprogramação com partições fixas.
 - Os que movem todo (ou parte do) processo entre a memória e o disco durante a sua execução:
 - Troca.
 - Memória virtual.
- A escolha do melhor sistema de gerenciamento depende de fatores como o custo, o tamanho, e a velocidade das memórias.
- Para a maior parte dos sistemas, o gerenciamento baseado em memória virtual é a melhor escolha.

Monoprogramação sem troca ou paginação

- Baseado em um gerenciamento de memória bem simples:
- O programa executa no processador até a sua conclusão:
 - O usuário digita um comando, e o sistema cria um processo para executar o programa deste comando.
 - Depois de o processo acabar a execução, o sistema mostra o prompt, e espera por outro comando do usuário.
 - A memória é compartilhada somente entre este programa e o sistema operacional:

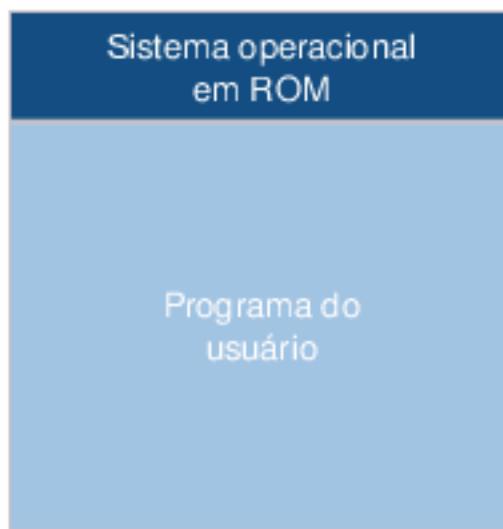


Nesta primeira organização, o sistema operacional é carregado dentro da parte inferior da memória RAM (Random Access Memory) quando o computador é inicializado, e o programa do usuário atualmente em execução pode usar toda a memória restante do sistema.

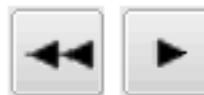


Monoprogramação sem troca ou paginação

- Baseado em um gerenciamento de memória bem simples:
- O programa executa no processador até a sua conclusão:
 - O usuário digita um comando, e o sistema cria um processo para executar o programa deste comando.
 - Depois de o processo acabar a execução, o sistema mostra o prompt, e espera por outro comando do usuário.
 - A memória é compartilhada somente entre este programa e o sistema operacional:



Na segunda organização, o sistema operacional está sempre armazenado dentro de uma memória ROM (Read-Only Memory), mapeada na parte superior da memória principal. Assim como antes, o programa do usuário pode usar toda a memória restante do sistema.



Monoprogramação sem troca ou paginação

- Baseado em um gerenciamento de memória bem simples:
- O programa executa no processador até a sua conclusão:
 - O usuário digita um comando, e o sistema cria um processo para executar o programa deste comando.
 - Depois de o processo acabar a execução, o sistema mostra o prompt, e espera por outro comando do usuário.
 - A memória é compartilhada somente entre este programa e o sistema operacional:



Na última organização, os drivers de dispositivo, que acessam os dispositivos, estão em uma memória ROM mapeada na parte superior da memória, e o sistema operacional é carregado na parte inferior da memória. O restante da memória pode ser usado pelo programa do usuário. Este modelo é o usado pelos micros IBM-PC, sendo que os drivers estão na BIOS, e o sistema operacional é, por exemplo, o MS-DOS.



Multiprogramação com partições fixas

- ➡ Gerenciamento usado pelos sistemas de gerenciamento em lote, como o sistema MFT usado pelo sistema OS/360.
- ➡ Pode ser também usado pelos sistemas interativos, pois nestes é comum que os processos bloqueiem por E/S.
- ➡ Permite a execução de vários trabalhos, do seguinte modo:
 - A memória é dividida em um conjunto de partições fixas, cujos tamanhos podem ser diferentes.
 - Ao executar um processo, o sistema o coloca em uma das partições disponíveis.
 - O processo ocupa uma partição até terminar a sua execução.



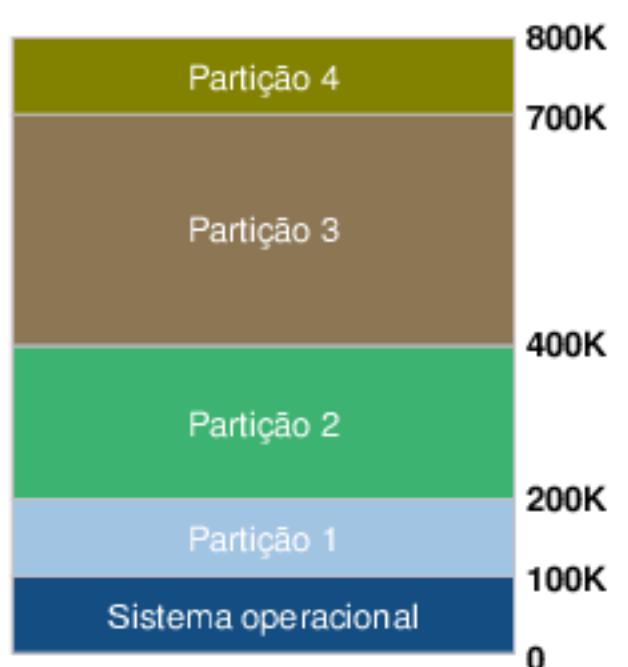
A memória é dividida entre o sistema operacional e um conjunto de n partições fixas, cujos tamanhos são definidos quando o sistema operacional é carregado. Os tamanhos destas partições podem ser diferentes, e não variam até que o sistema seja reiniciado.

Multiprogramação com partições fixas



Um modo de escolher uma partição é o de associar uma fila a cada uma das partições:

- O trabalho é colocado na fila cuja partição possui o tamanho mais próximo do tamanho do processo.
- Quando uma partição for liberada, o primeiro trabalho da fila associado à partição é o escolhido.



Suponha que o sistema operacional inicializou, e que foram criadas quatro partições, nas quais poderão ser colocados os processos que são usados para executar os trabalhos submetidos ao sistema. Os tamanhos das partições são, como podemos ver pela figura, iguais a 100K, 200K, 300K e 100K. O sistema operacional está numa partição de 100K.

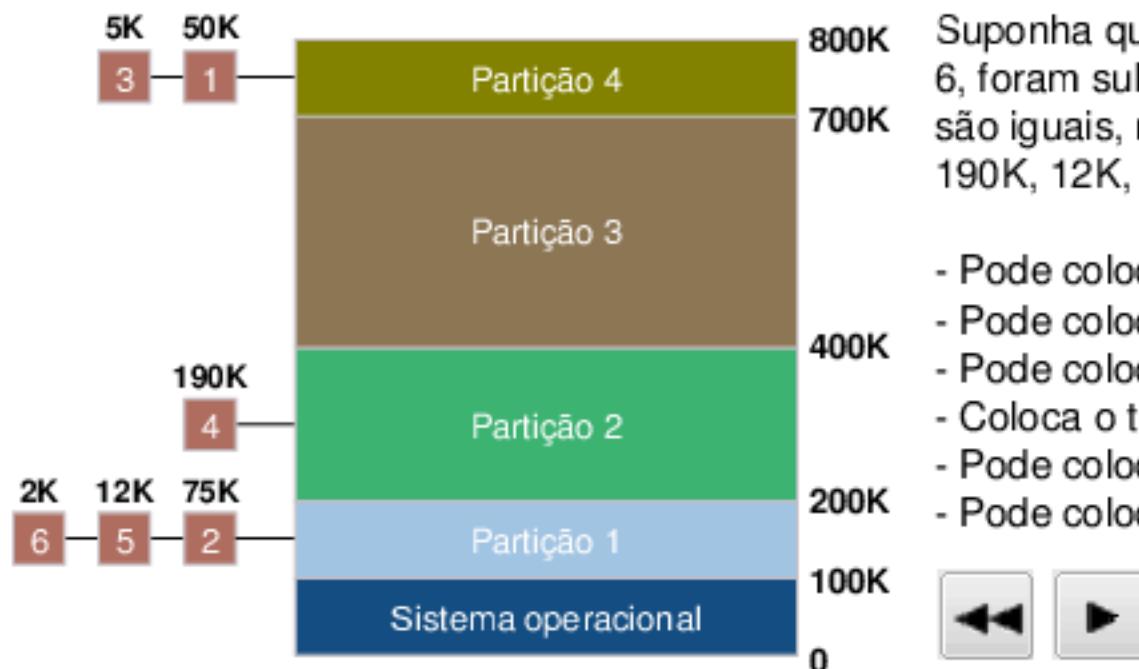


Multiprogramação com partições fixas



Um modo de escolher uma partição é o de associar uma fila a cada uma das partições:

- O trabalho é colocado na fila cuja partição possui o tamanho mais próximo do tamanho do processo.
- Quando uma partição for liberada, o primeiro trabalho da fila associado à partição é o escolhido.



Suponha que seis trabalhos, numerados de 1 a 6, foram submetidos, sendo que seus tamanhos são iguais, respectivamente, a 50K, 75K, 5K, 190K, 12K, e 2K. Então, o sistema:

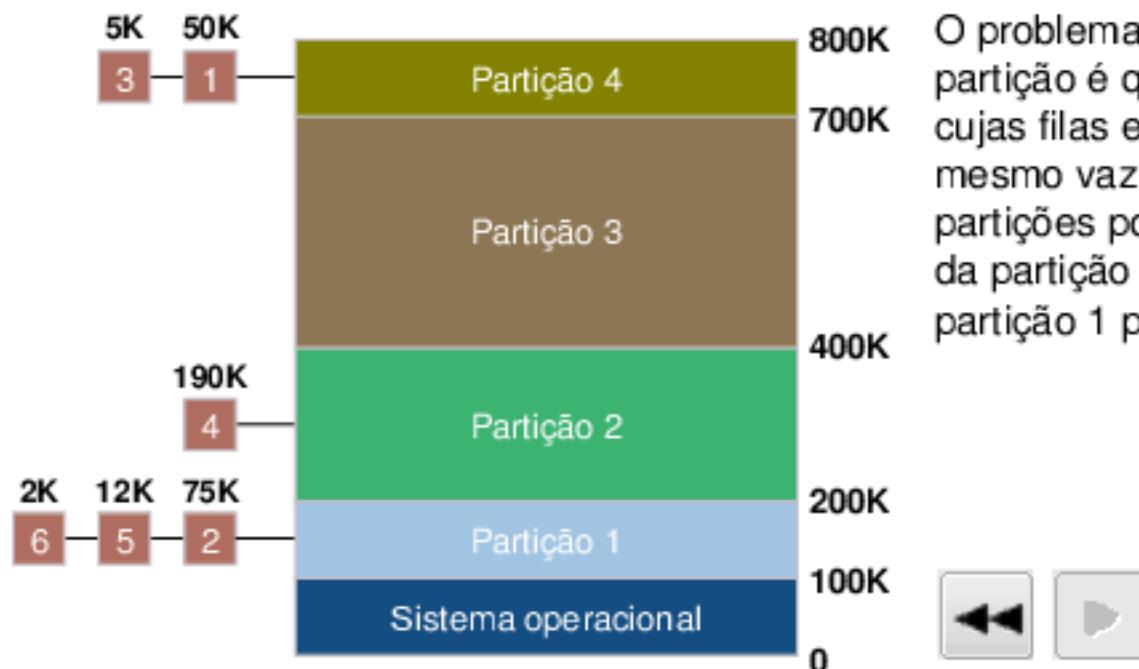
- Pode colocar o trabalho 1 na fila da partição 4.
- Pode colocar o trabalho 2 na fila da partição 1.
- Pode colocar o trabalho 3 na fila da partição 4.
- Coloca o trabalho 4 na fila da partição 2.
- Pode colocar o trabalho 5 na fila da partição 1.
- Pode colocar o trabalho 6 na fila da partição 1.

Multiprogramação com partções fixas



Um modo de escolher uma partição é o de associar uma fila a cada uma das partções:

- O trabalho é colocado na fila cuja partição possui o tamanho mais próximo do tamanho do processo.
- Quando uma partição for liberada, o primeiro trabalho da fila associado à partição é o escolhido.



O problema com este modo de escolha de uma partição é que podemos ter algumas partções cujas filas estão com poucos trabalhos, ou até mesmo vazias, enquanto que as filas das outras partções podem estar cheias. No exemplo, a fila da partição 3 está vazia, enquanto que a fila da partição 1 possui três processos.

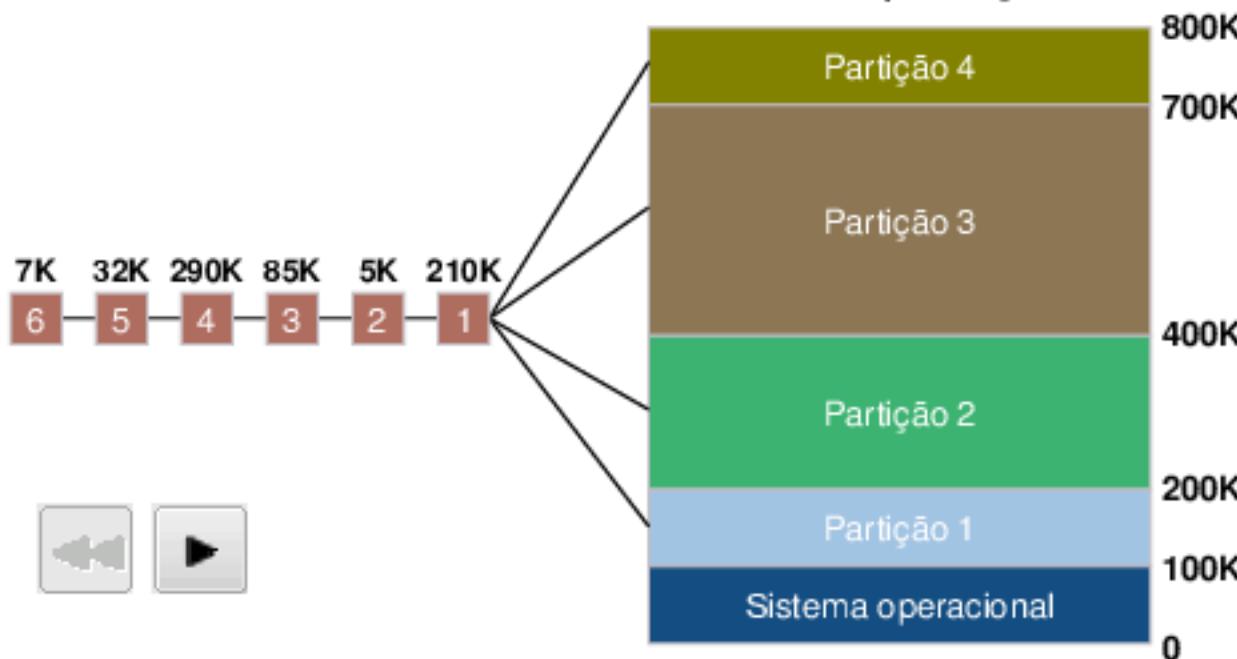


Multiprogramação com partícões fixas



Uma outra idéia é a de termos uma única fila:

- Todos os trabalhos são colocados nesta fila.
- Quando uma partição for liberada, podemos:
 - Ou escolher o trabalho mais próximo do inicio da fila cujo tamanho é menor ou igual ao desta partição.
 - Ou escolher o trabalho da fila cujo tamanho é o mais próximo do tamanho desta partição.



O sistema criou quatro partícões que podem ser usadas para a execução dos trabalhos, com tamanhos iguais a 100K, 200K, 300K e 100K. Agora, temos uma única fila com todos os trabalhos. Suponha que seis trabalhos foram submetidos ao sistema, cujos tamanhos são dados acima destes trabalhos.

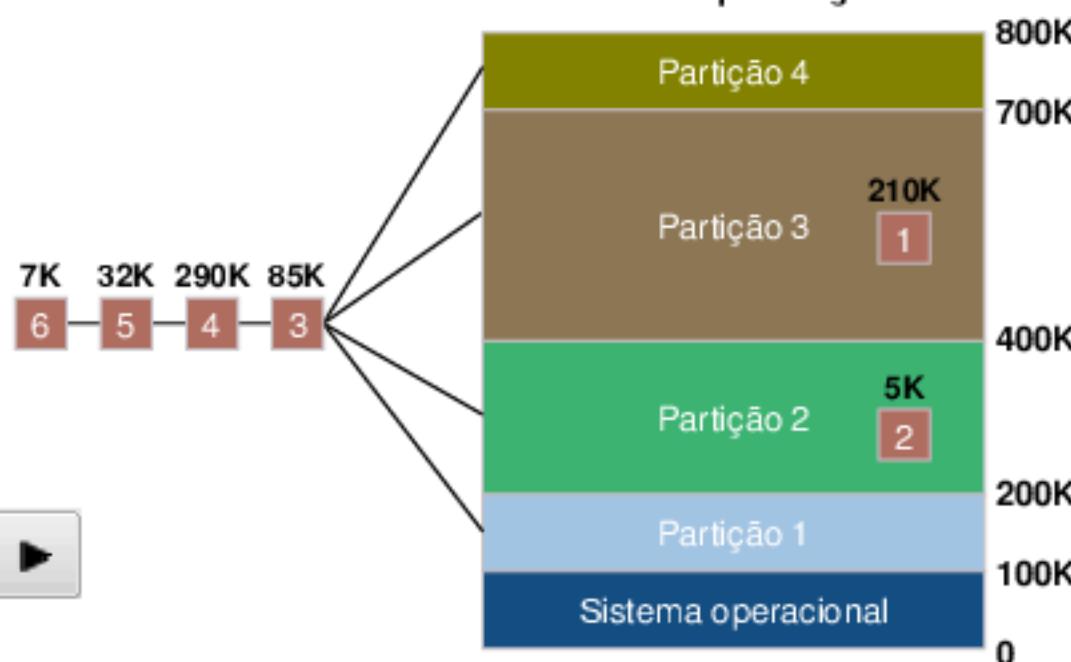


Multiprogramação com partções fixas



Uma outra idéia é a de termos uma única fila:

- Todos os trabalhos são colocados nesta fila.
- Quando uma partição for liberada, podemos:
 - Ou escolher o trabalho mais próximo do inicio da fila cujo tamanho é menor ou igual ao desta partição.
 - Ou escolher o trabalho da fila cujo tamanho é o mais próximo do tamanho desta partição.



Um primeiro algoritmo poderia ser o seguinte: quando uma das partções é liberada, o primeiro trabalho da fila cujo tamanho é menor do que o da partição é o escolhido. Por exemplo, se a partição 2 for liberada, o trabalho 2 será escolhido e colocado na partição. Depois disso, se a partição 3 for liberada, o trabalho 1 será o escolhido.

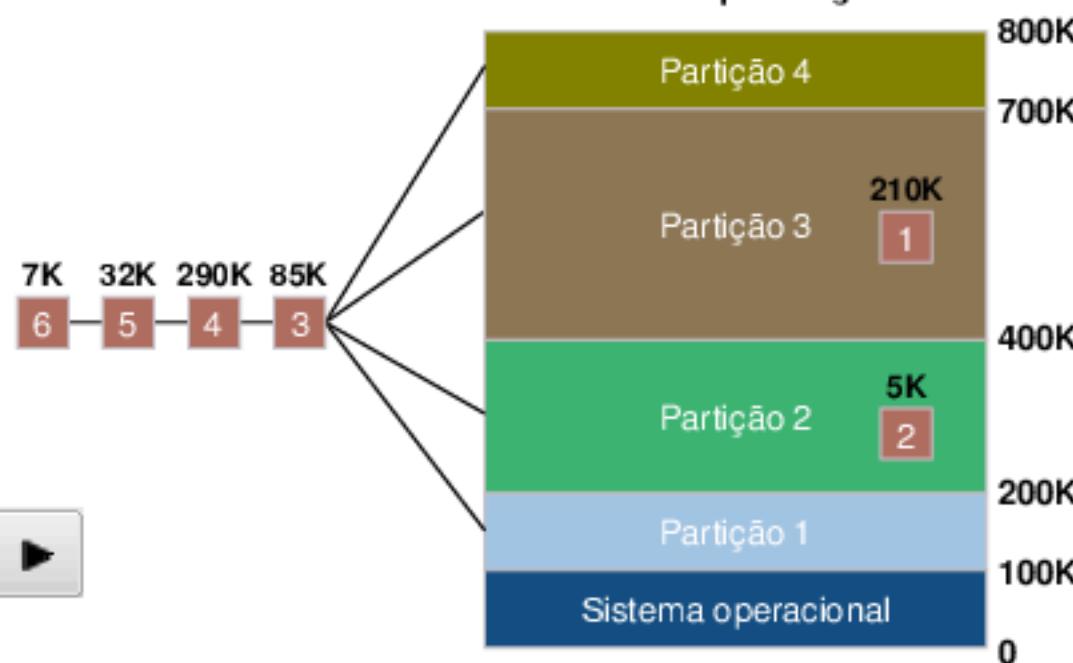


Multiprogramação com partições fixas



Uma outra idéia é a de termos uma única fila:

- Todos os trabalhos são colocados nesta fila.
- Quando uma partição for liberada, podemos:
 - Ou escolher o trabalho mais próximo do inicio da fila cujo tamanho é menor ou igual ao desta partição.
 - Ou escolher o trabalho da fila cujo tamanho é o mais próximo do tamanho desta partição.



O problema deste algoritmo é o desperdício de espaço dentro da partição: na partição 2, o espaço desperdiçado foi de 195K, e na partição 3, o espaço gasto foi de 90K.

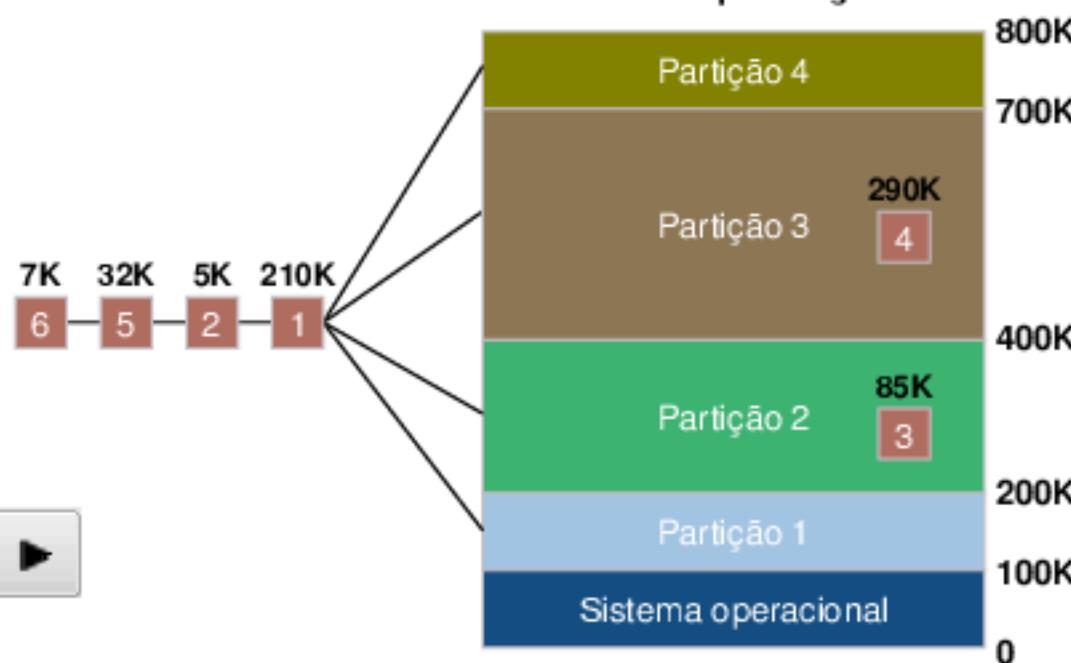


Multiprogramação com partícões fixas



Uma outra idéia é a de termos uma única fila:

- Todos os trabalhos são colocados nesta fila.
- Quando uma partição for liberada, podemos:
 - Ou escolher o trabalho mais próximo do inicio da fila cujo tamanho é menor ou igual ao desta partição.
 - Ou escolher o trabalho da fila cujo tamanho é o mais próximo do tamanho desta partição.



Um outro algoritmo seria o de escolher o processo da fila cujo tamanho mais se aproxima do tamanho da partição que foi liberada. Quando a partição 2 for liberada, o processo 3 será o escolhido, pois o tamanho da partição é de 100K. Se, depois disso, a partição 3 for liberada, o processo 4 será o escolhido.

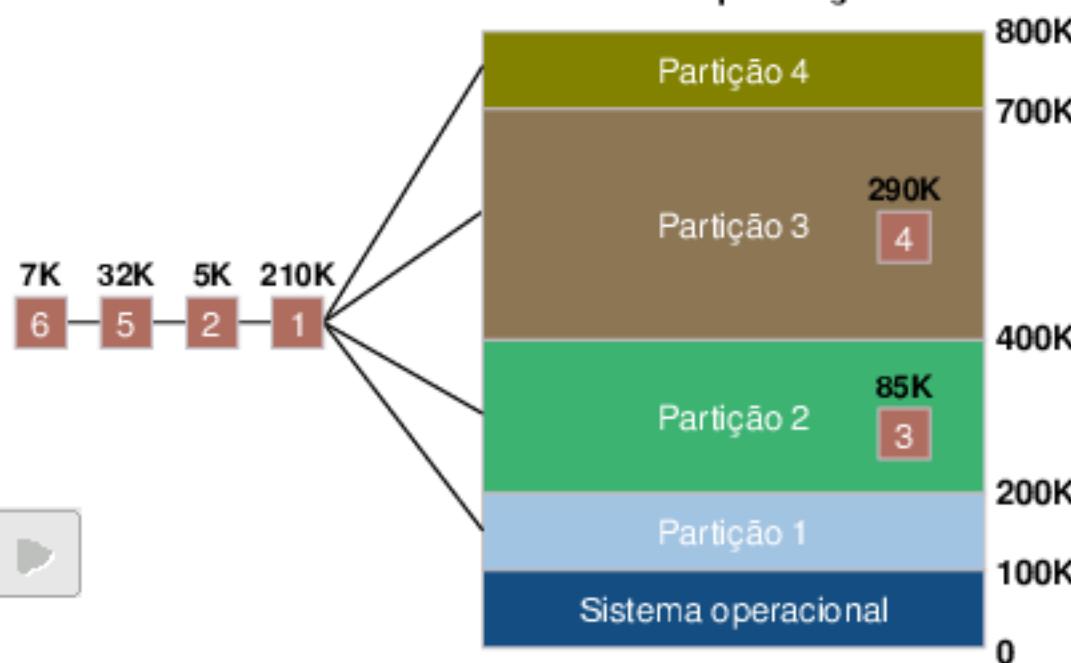


Multiprogramação com partções fixas



Uma outra idéia é a de termos uma única fila:

- Todos os trabalhos são colocados nesta fila.
- Quando uma partição for liberada, podemos:
 - Ou escolher o trabalho mais próximo do inicio da fila cujo tamanho é menor ou igual ao desta partição.
 - Ou escolher o trabalho da fila cujo tamanho é o mais próximo do tamanho desta partição.

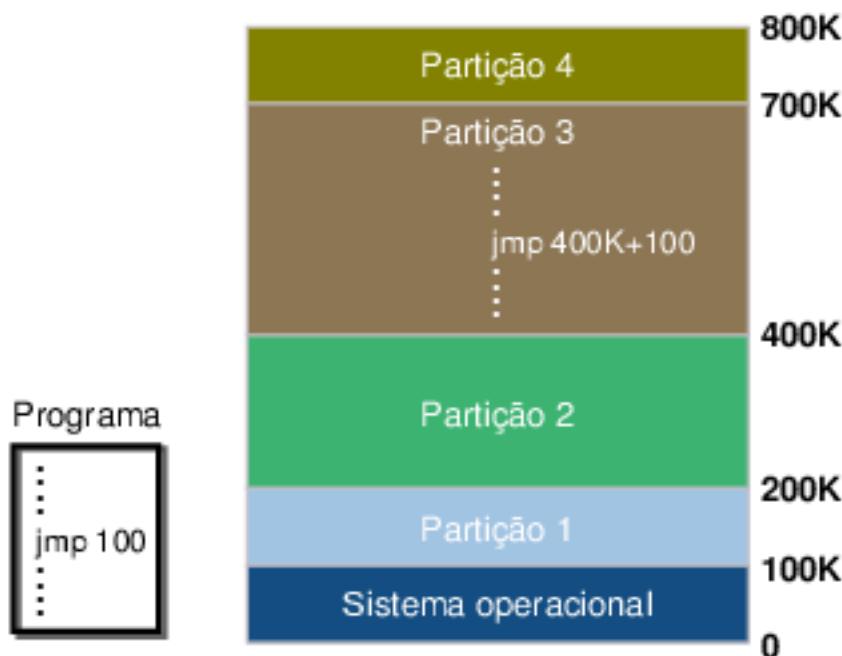


Ao usarmos este algoritmo, os espaços gastos dentro das partções serão os menores possíveis. O problema é que os trabalhos menores podem nunca ser executados, como, por exemplo, o trabalho 5, caso sempre seja submetido um outro trabalho com tamanho maior do que 5K, antes de este trabalho ser executado.



Realocação e proteção

- O uso da multiprogramação introduz os seguintes problemas:
- **Realocação:** garantir que as instruções de acesso à memória irão acessar os endereços corretos:



Os endereços de um programa são determinados durante a vinculação do programa. Por exemplo, se um programa foi compilado a partir do endereço 0, e se for carregado a partir do endereço 400K (na partição 3), o programa fará uma chamada dentro do sistema operacional, se possuir, por exemplo, uma instrução de salto para o endereço 100. Neste caso, deveremos somar 400K a todos os endereços.

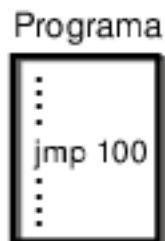
- **Proteção:** garantir que um programa somente acesse a região da memória que foi alocada a ele.

Realocação e proteção

Os problemas da realocação e da proteção podem ser resolvidos pelo hardware, usando dois registradores especiais:

- **Registrador de base (RB)**: contém o endereço inicial de uma partição. Todo o endereçamento é relativo a este registrador.
- **Registrador de limite (RL)**: contém o tamanho da partição. Os endereços não podem ser maiores do que este limite.
- O hardware informa ao sistema qualquer tentativa, por um processo, de acessar uma área da memória não permitida.

RB = 400K
RL = 300K

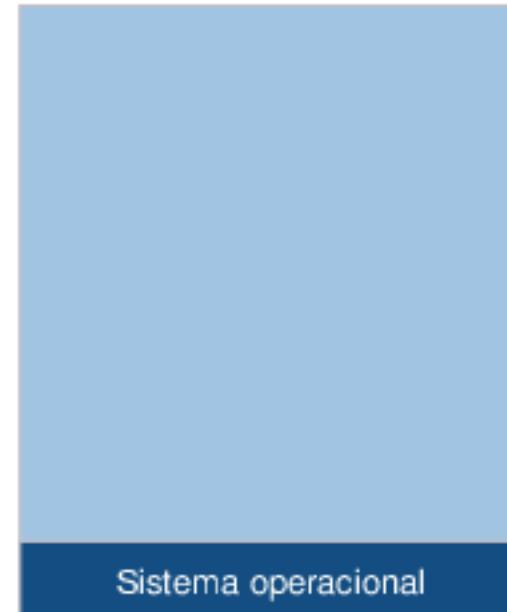


Quando estes registradores estão disponíveis, todos os programas podem ser compilados a partir do endereço 0, e, quando carregarmos um programa em uma das partições, basta alterar o valor de RB para o início da partição, e valor de RL para o tamanho da partição.

No exemplo anterior, basta o sistema copiar o valor 400K para RB, e copiar 300K, o tamanho da partição, para RL. Como os endereços são relativos a BP, a instrução de salto jmp 100 saltará para o endereço 400K+100.

Troca

- Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:
 - A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
 - Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



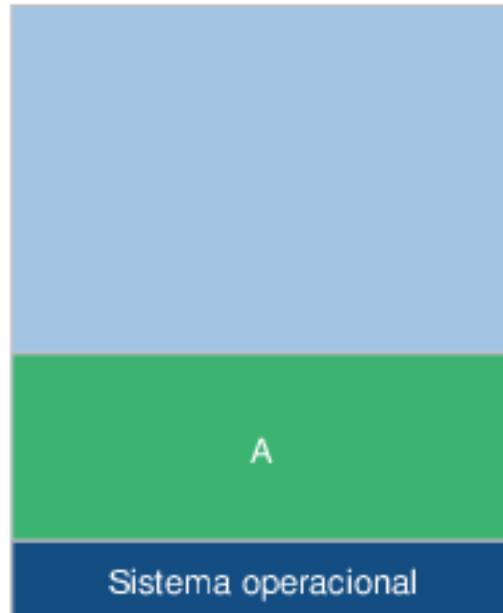
Inicialmente, a memória está vazia, e somente o sistema operacional está carregado na memória do computador.

Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.

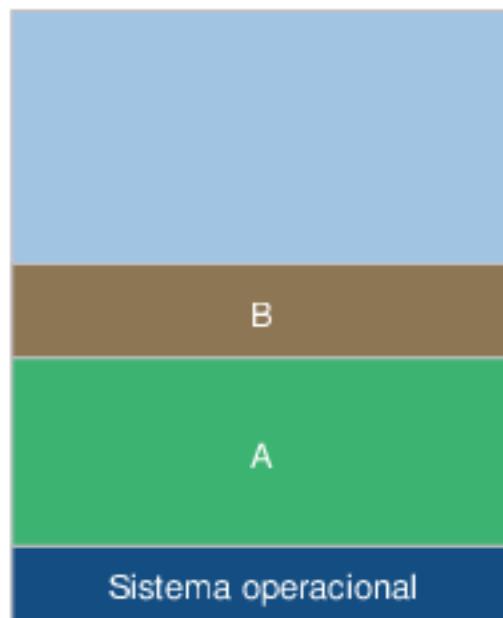


Agora, o processo A foi iniciado pelo sistema. Com isso, o sistema criou uma partição na memória com o tamanho do processo A, e carregou o processo nesta partição, para que este possa ser executado.

Memória livre

Troca

- Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:
 - A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
 - Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



Mais um processo, o B, foi também iniciado, e uma nova partição com o seu tamanho foi criada na memória.



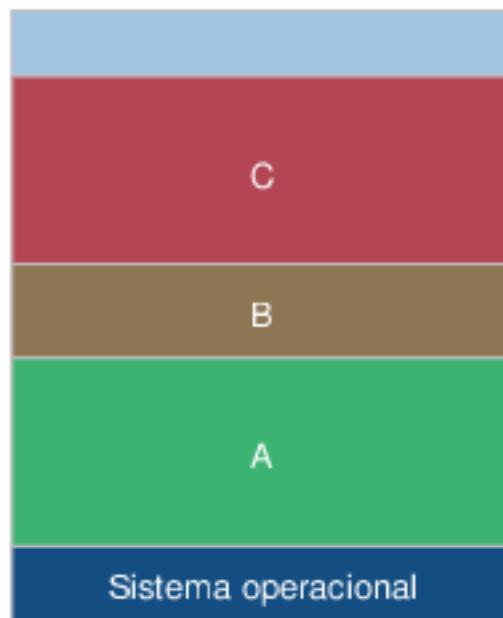
Memória livre

Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



Mais um processo, o C, foi carregado na memória, e mais uma partição, com o tamanho de C, foi criada.



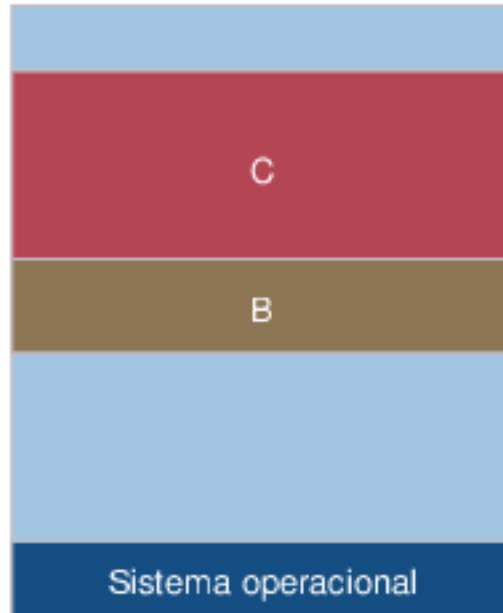
Memória livre

Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



O processo A foi retirado da memória, porque terminou a sua execução, ou porque foi salvo no disco, pois A já ficou muito tempo na memória. Com isso, a parte da memória alocada à partição foi liberada para ser usada para criar outras partições.

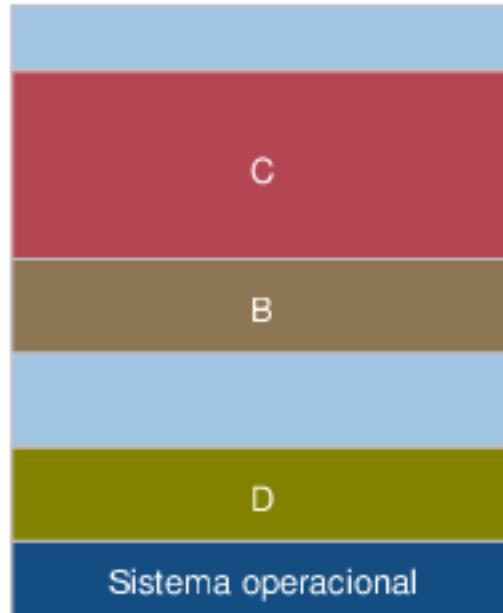


Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



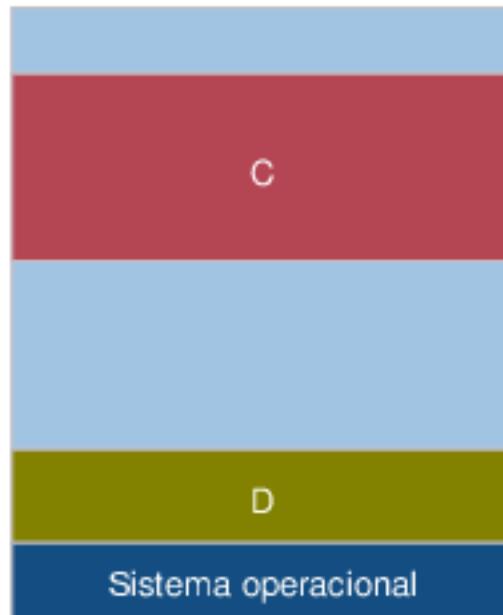
Agora, um novo processo, o D, foi criado, e com isso, o sistema criou uma nova partição. Esta partição foi criada na posição da memória em que o processo A tinha sido carregado, e como o tamanho de D é menor do que o de A, uma parte da memória liberada ao mover A para o disco não será usada ao alojar a nova partição.

Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.



Agora, o processo B ou terminou a sua execução, ou foi salvo no disco. A parte da memória ocupada pela sua partição foi então liberada.

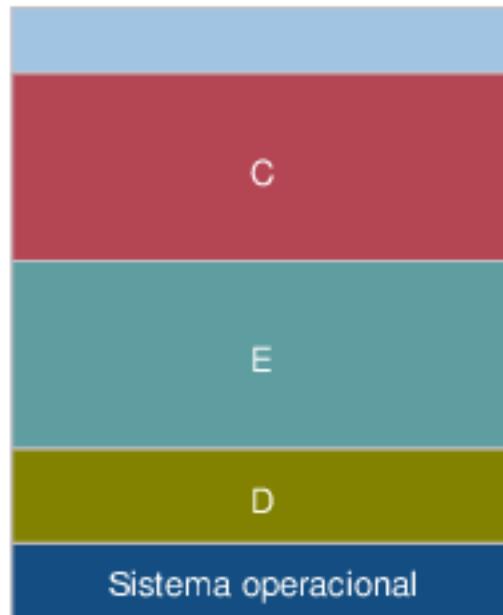


Troca



Gerenciamento de memória usado quando a memória não pode armazenar todos os processos do sistema:

- A memória também é particionada, mas o número, o tamanho, e a posição das partições varia dinamicamente.
- Quando o escalonador escolhe um processo:
 - O processo em execução é copiado para o disco, e a partição usada pelo processo é liberada.
 - Uma nova partição é criada na qual será copiado, do disco, o novo processo escolhido pelo escalonador.

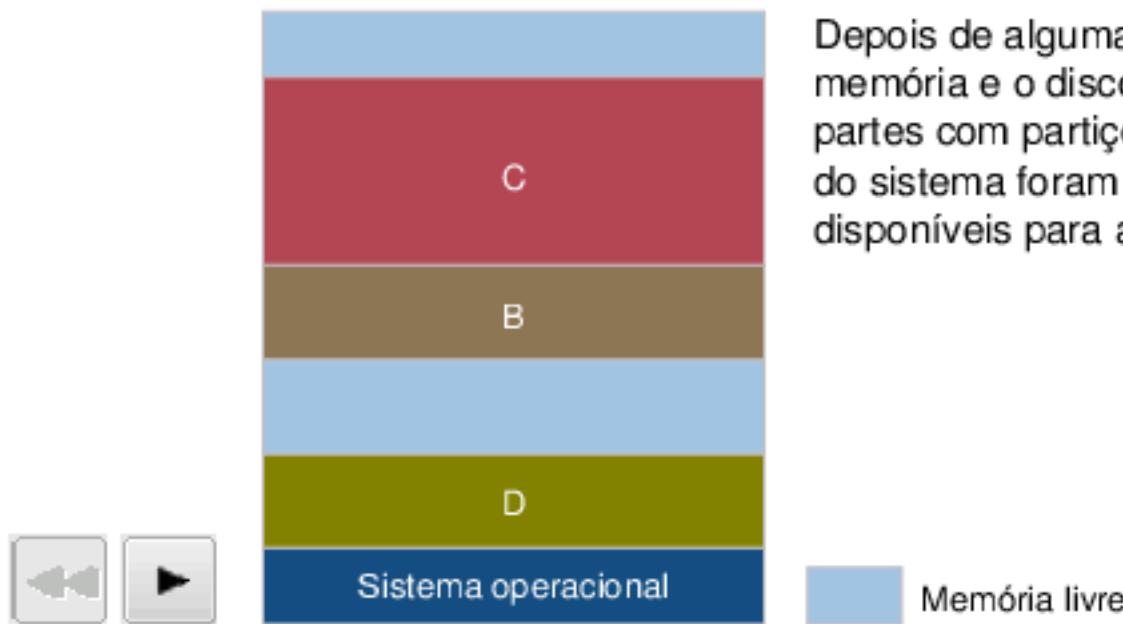


Agora um novo processo, o E, é criado ou lido do disco. A partição alocada ao processo ocupa toda a área de memória disponível entre as partições dos processos D e C.



Troca

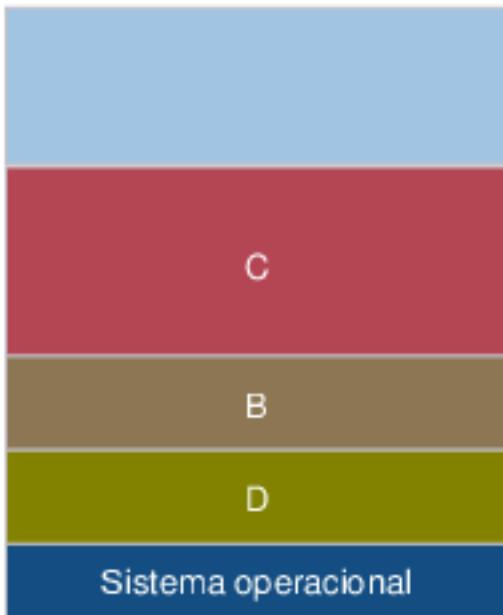
- ➡ A grande vantagem da troca sobre o uso de partições fixas é a de que otimizamos o uso da memória:
 - As partições possuem o tamanho correto do processo, e com isso, não há desperdício de espaço dentro da partição.
- ➡ Porém, o gerenciamento da memória será mais complicado, pois deveremos dinamicamente alocar e desalocar as partições.
- ➡ Um outro problema é que esta alocação e desalocação dinâmica pode causar a **fragmentação da memória**:



Depois de algumas trocas de processos entre a memória e o disco, a memória fica dividida entre partes com partições, em que alguns processos do sistema foram carregados, e partes livres, disponíveis para a criação de novas partições.

Troca

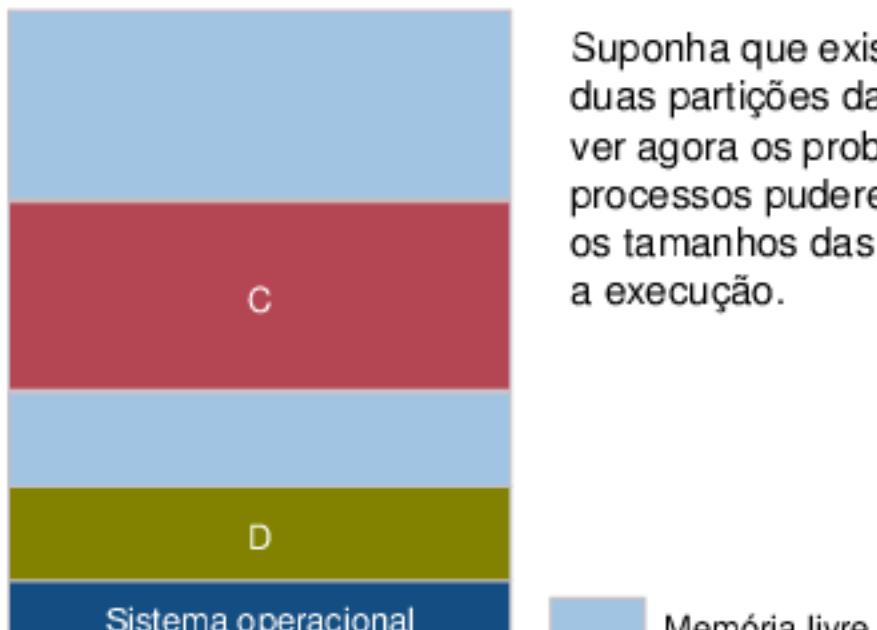
- ➡ A grande vantagem da troca sobre o uso de partições fixas é a de que otimizamos o uso da memória:
 - As partições possuem o tamanho correto do processo, e com isso, não há desperdício de espaço dentro da partição.
- ➡ Porém, o gerenciamento da memória será mais complicado, pois deveremos dinamicamente alocar e desalocar as partições.
- ➡ Um outro problema é que esta alocação e desalocação dinâmica pode causar a **fragmentação da memória**:



Quando as áreas disponíveis na memória são pequenas e não contíguas, podemos **compactar a memória**: as partições dos programas são deslocadas para os endereços mais baixos da memória, até que a memória seja dividida em duas grandes áreas - uma com todas as partições criadas, e uma outra com toda a área disponível da memória. O problema é que o processo de compactação é muito custoso.

Troca

- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.

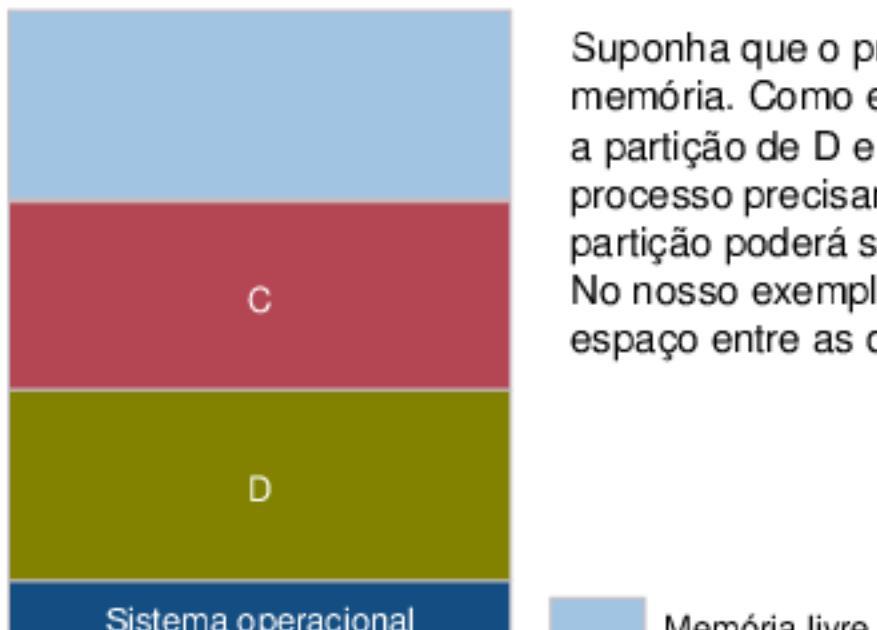


Suponha que existem dois processos, C e D, em duas partições da memória não contíguas. Vamos ver agora os problemas que podem ocorrer se os processos puderem aumentar de tamanho, isto é, os tamanhos das partições podem variar durante a execução.



Troca

- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.

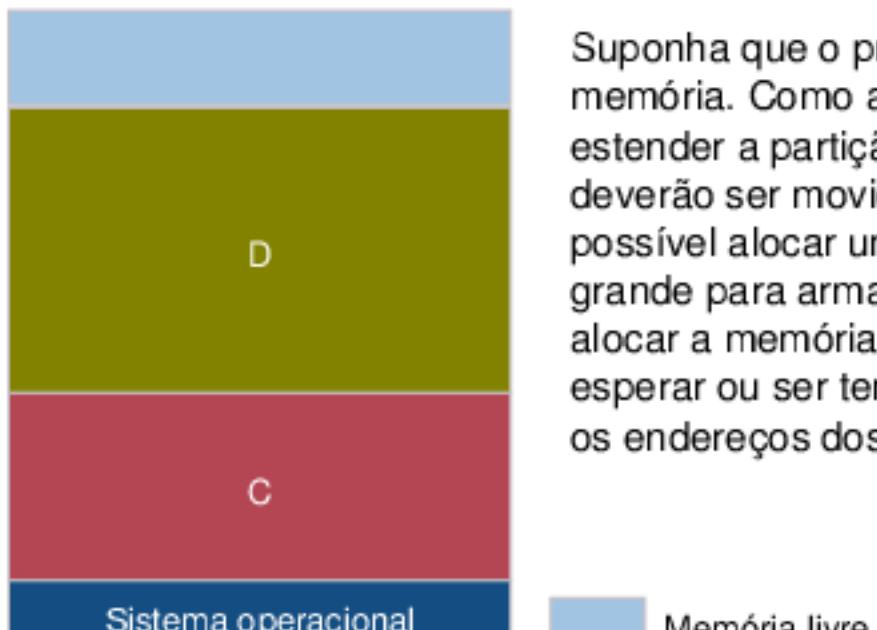


Suponha que o processo D precisou de mais memória. Como existe espaço disponível entre a partição de D e a partição de C, então, se o processo precisar no máximo deste espaço, a partição poderá ser estendida sem problemas. No nosso exemplo, o processo D alocou todo o espaço entre as duas partições.



Troca

- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.



Suponha que o processo D precisou alocar mais memória. Como agora não existe espaço para estender a partição, então um ou mais processos deverão ser movidos na memória, para que seja possível alocar uma partição suficientemente grande para armazenar D. Se não for possível alocar a memória, então o processo deverá esperar ou ser terminado. No exemplo, trocamos os endereços dos processos C e D na memória.



Troca

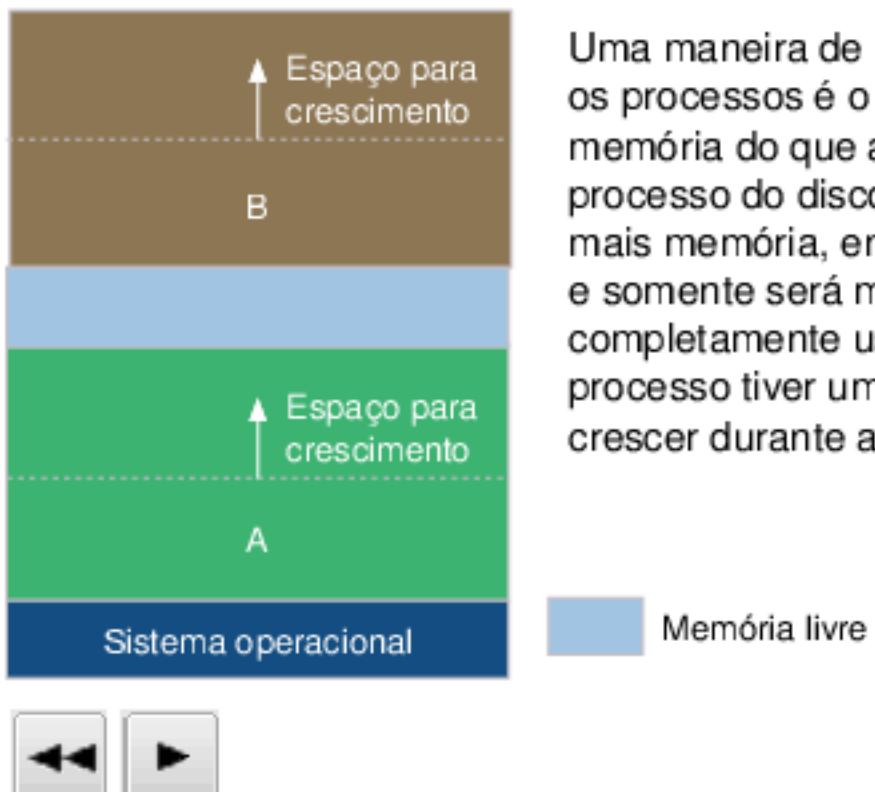
- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.



Suponha que um processo B foi carregado na última área livre da memória. Agora, não temos nenhuma memória disponível para estender ou para criar novas partições. Se, por exemplo, o processo C desejar alocar mais memória, este ou deverá esperar até que exista memória disponível, ou deverá ser terminado.

Troca

- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.



Uma maneira de minimizar o tempo gasto ao mover os processos é o de simplesmente alocar mais memória do que a necessária para copiar um processo do disco. Se um processo precisar de mais memória, então este usa a memória adicional, e somente será movido se a memória adicional for completamente usada. Esta organização é útil se o processo tiver uma única área de dados que pode crescer durante a execução.

Troca

- Um detalhe é o de quanta memória será alocada a um processo:
- Se o tamanho do processo é fixo, então o espaço alocado será igual a este tamanho.
 - Se o tamanho do processo variar durante a execução, então deveremos alocar mais memória para este processo.



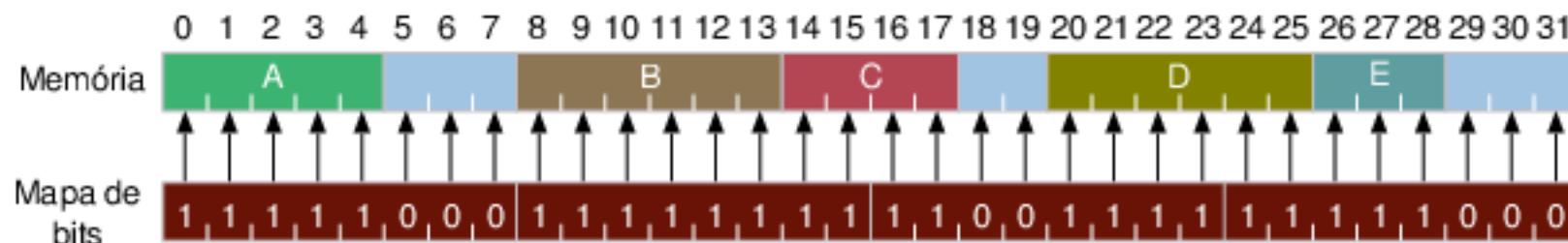
Se a partição deve ser dividida em uma região com o programa, que ocupa o início da partição, uma região com os dados, que começa logo após o programa, e que cresce em direção ao final da partição, e uma região de pilha, que cresce em direção ao início da partição, e que fica no final da partição, então podemos alocar uma partição com mais espaço, dividido de acordo com a figura. Entre as regiões da pilha e de dados, existe uma região livre, que pode ser usada para estender ambas as regiões, se for necessário.

Memória livre



Troca

- A memória pode ser gerenciada através de um **mapa de bits**:
- A memória é dividida em blocos de tamanho fixo, chamados de **unidades de alocação**.
 - Existe um bit no mapa para cada unidade de alocação, que é igual a 1 se a unidade está sendo usada, e 0 em caso contrário.

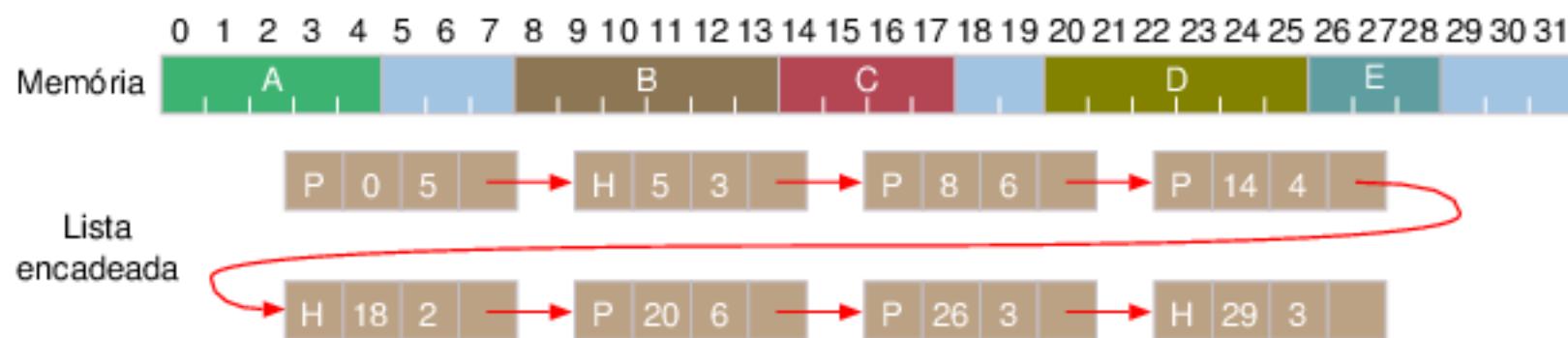


Na figura, mostramos as 32 unidades de alocação iniciais da memória. Para cada uma destas unidades, existe um bit no mapa de bits, que indica se a unidade está ou não disponível.

- O tamanho da unidade de alocação é importante:
 - O tamanho do mapa de bits depende deste tamanho.
 - O tempo necessário para procurar por um conjunto destas unidades depende do tamanho do mapa de bits.

Troca

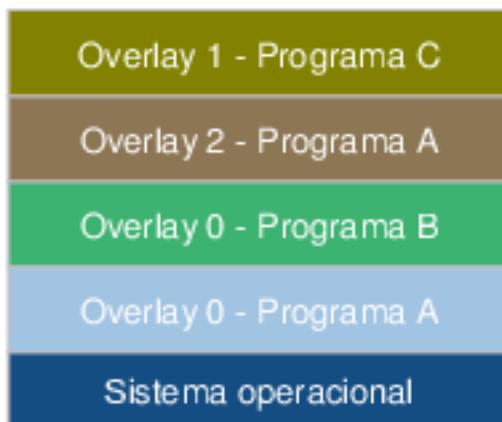
- ➡ O grande problema do uso do mapa de bits é o tempo gasto ao alocar uma partição para um processo copiado do disco:
 - Se um processo ocupa k unidades de alocação, deveremos varrer o mapa até achar k unidades consecutivas disponíveis.
- ➡ Uma outra solução é o gerenciamento da memória através de **listas encadeadas**, sendo que cada elemento da lista:
 - Representa um conjunto consecutivo de unidades de alocação.
 - Contém o endereço inicial do conjunto, o seu tamanho, e o seu tipo, e um ponteiro para o próximo elemento.



Lista encadeada para o exemplo anterior, para as 32 unidades de alocação iniciais da memória. O tipo P indica que as unidades estão alocadas a um processo, e o H indica que as unidades estão disponíveis.

Memória virtual

- ➡ Nem todos os programas que são executados em um computador podem ser totalmente armazenados na memória.
- ➡ Nestes casos, podemos usar o conceito de overlays:
 - O programa é dividido em partes menores que são chamadas de **overlays**.
 - O overlay 0 é o primeiro a ser executado.
 - Quando necessário, um overlay chama o outro overlay, que será carregado na memória pelo sistema.
 - O sistema gerencia a troca dos overlays entre a memória e o disco, e permite que múltiplos overlays estejam na memória.

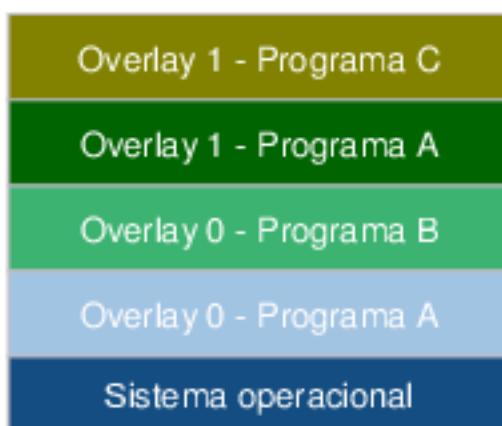


Neste conceito, os programas são divididos em partes menores, chamadas de overlays, pelo programador. No exemplo, temos quatro overlays na memória: os overlays 0 e 2 do programa A, o overlay 0 do programa B, e o overlay 1 do programa C.



Memória virtual

- ➡ Nem todos os programas que são executados em um computador podem ser totalmente armazenados na memória.
- ➡ Nestes casos, podemos usar o conceito de overlays:
 - O programa é dividido em partes menores que são chamadas de **overlays**.
 - O overlay 0 é o primeiro a ser executado.
 - Quando necessário, um overlay chama o outro overlay, que será carregado na memória pelo sistema.
 - O sistema gerencia a troca dos overlays entre a memória e o disco, e permite que múltiplos overlays estejam na memória.



Se o overlay 0 do programa A chamar o overlay 1 deste programa, então algum outro overlay será salvo no disco, antes deste overlay ser carregado. Suponha que o overlay 2 de A foi o escolhido para ser salvo no disco. Então, o overlay 1 de A será carregado na posição da memória do overlay 2, e será executado.



Memória virtual

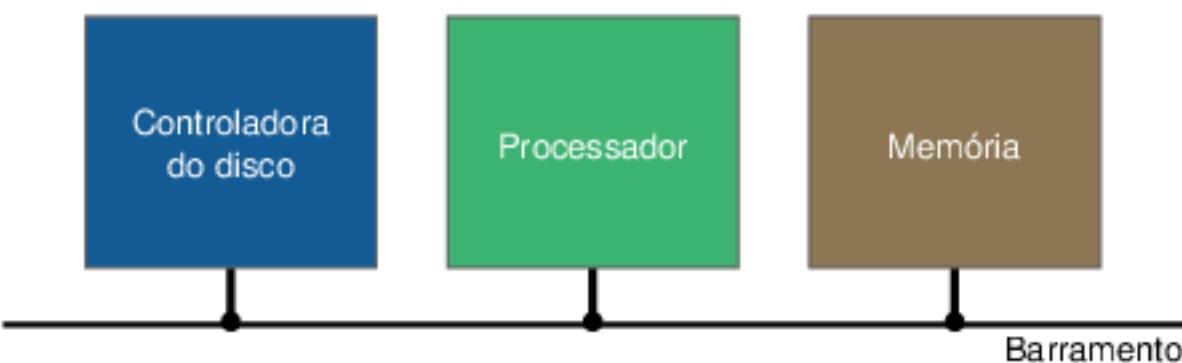
- ➡ O problema do uso de overlays é a complexidade na programação dos programas, pois a divisão é feita pelo programador.
- ➡ Uma outra possibilidade é o uso do conceito de **memória virtual**, criado por Fotheringham em 1961:
 - O tamanho do espaço de endereçamento do processo que executa o programa pode ser maior do que a memória.
 - As partes atualmente usadas do espaço de endereçamento são mantidas na memória, e as restantes, no disco.
 - Ao contrário dos overlays, o sistema é que é responsável pela divisão do espaço de endereçamento.



Os programas continuam sendo divididos em vários pedaços, mas agora o sistema operacional gerencia a troca das partes entre o disco e a memória, e a divisão do programa nestas partes. Por exemplo, o sistema pode executar um programa de 16MB em uma memória com 4MB, se dividir o programa em quatro partes de 4MB, e escolher qual dos 4MB deve estar na memória em cada instante.

Memória virtual

- ➡ Algumas das instruções do processador acessam endereços da memória, que são chamados de **endereços virtuais**.
- ➡ O conjunto de endereços que um programa pode acessar através das instruções é chamado de **espaço de endereçamento virtual**.
- ➡ Quando não usamos o conceito de memória virtual:
 - Os endereços são copiados diretamente no barramento da memória pelo processador.
 - O endereço físico lido (ou escrito) da (na) memória é o mesmo endereço virtual gerado pela instrução.

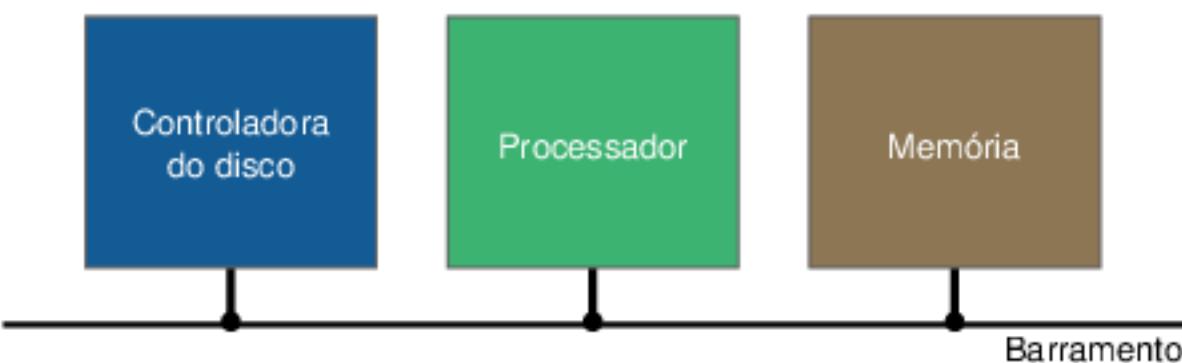


Quando o processador executa uma instrução, o endereço usado para obter o seu código é enviado diretamente para a memória.



Memória virtual

- ➡ Algumas das instruções do processador acessam endereços da memória, que são chamados de **endereços virtuais**.
- ➡ O conjunto de endereços que um programa pode acessar através das instruções é chamado de **espaço de endereçamento virtual**.
- ➡ Quando não usamos o conceito de memória virtual:
 - Os endereços são copiados diretamente no barramento da memória pelo processador.
 - O endereço físico lido (ou escrito) da (na) memória é o mesmo endereço virtual gerado pela instrução.



Por exemplo, se o processador deseja executar a instrução cujo código começa no endereço FEDC, o processador envia este endereço diretamente para o barramento da memória, e obtém\ o código FE da instrução.

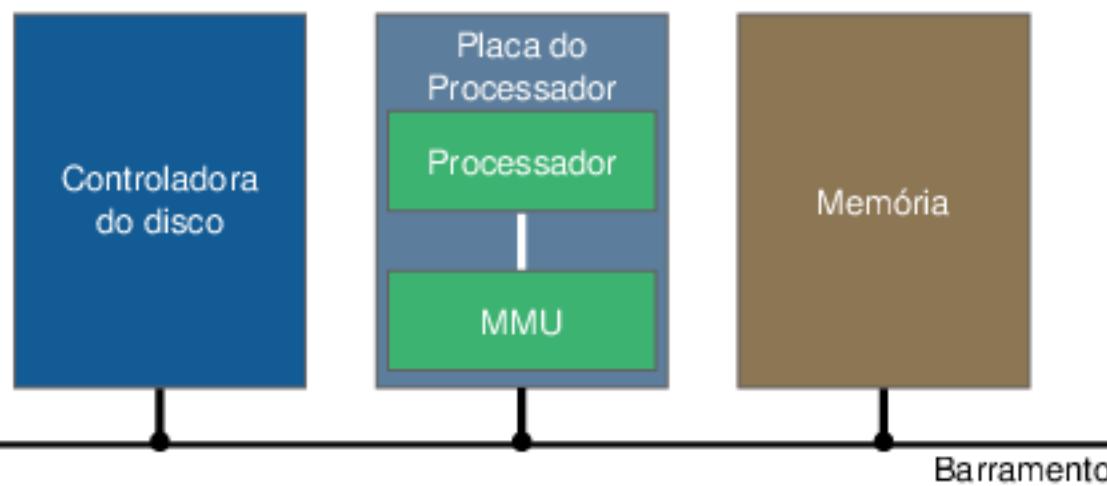


Memória virtual



Quando usamos o conceito de memória virtual:

- Os endereços são passados a uma unidade especial, chamada de MMU (Memory Management Unit).
- A MMU converte o endereço virtual em um endereço físico da memória, que em geral é diferente deste endereço virtual.
- Depois da conversão, a MMU copia o endereço físico gerado no barramento da memória.



No conceito de memória virtual, quando o processador executa uma instrução, o endereço virtual usado para obter o seu código é enviado para a MMU, que então o converte para o endereço físico com o código da instrução.

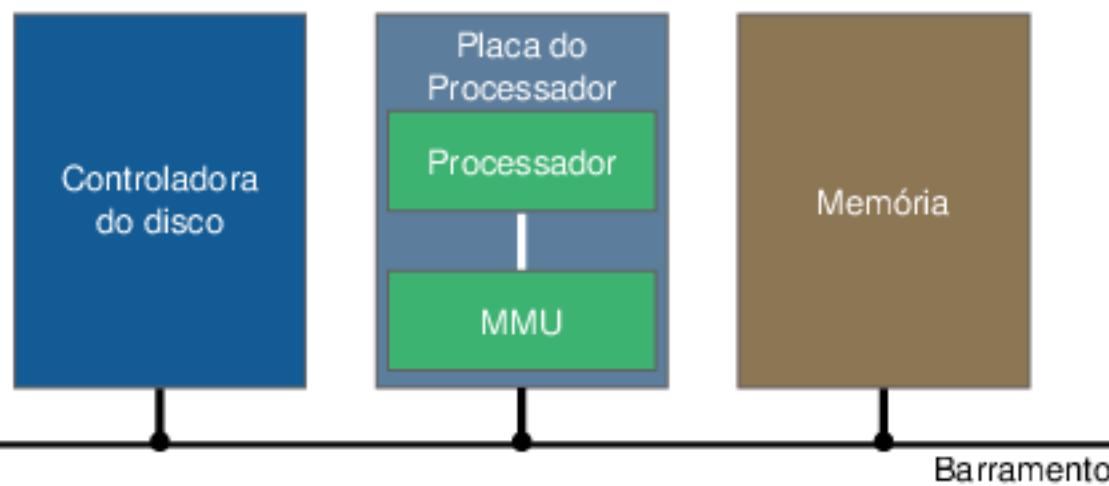


Memória virtual



Quando usamos o conceito de memória virtual:

- Os endereços são passados a uma unidade especial, chamada de MMU (Memory Management Unit).
- A MMU converte o endereço virtual em um endereço físico da memória, que em geral é diferente deste endereço virtual.
- Depois da conversão, a MMU copia o endereço físico gerado no barramento da memória.

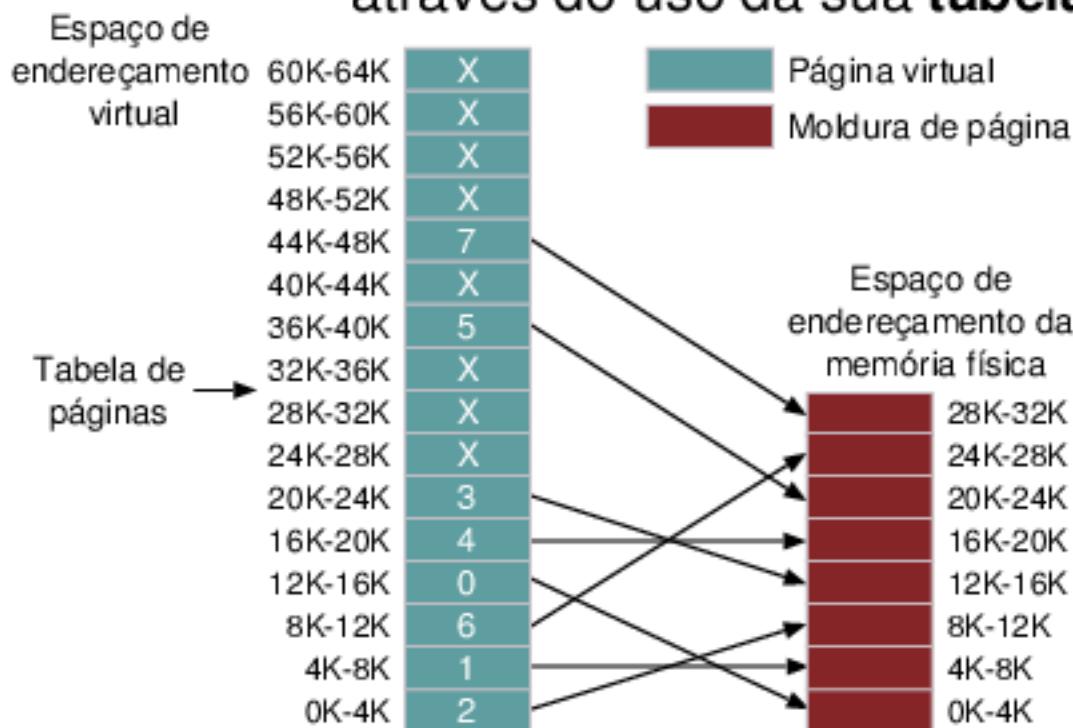


Por exemplo, se o processador deseja executar a instrução cujo código começa no endereço virtual AADC, o processador envia este endereço para a MMU, que o converte no endereço físico 80DC com o código FE da instrução



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

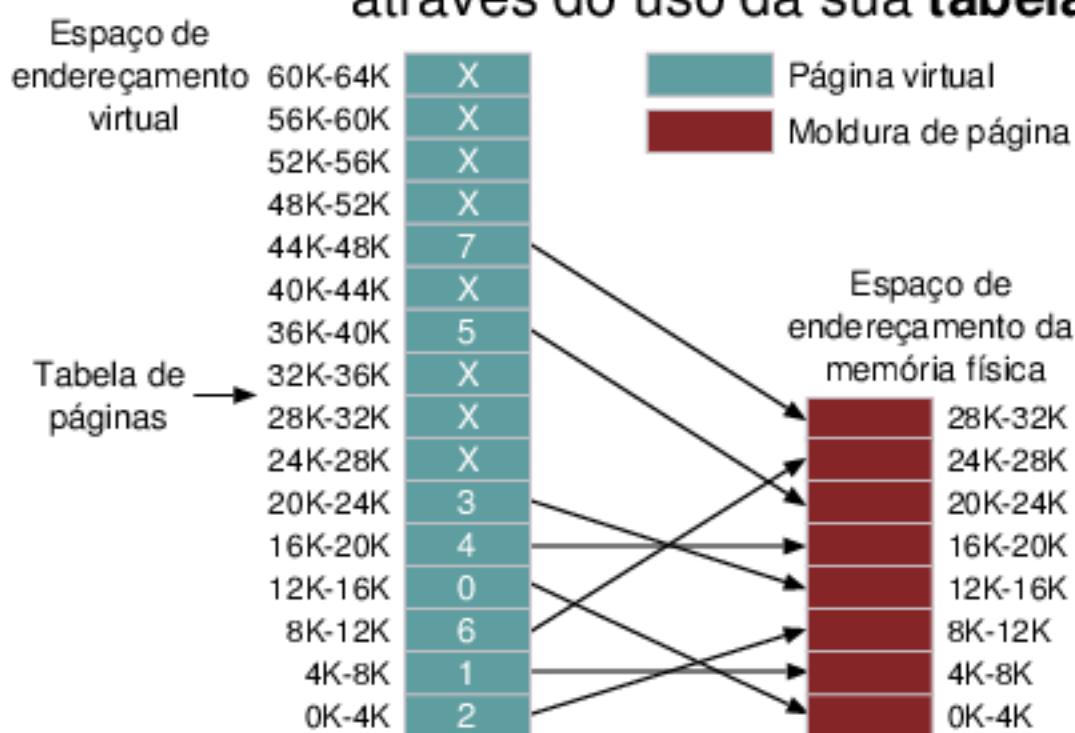


Exemplo de um processador com um espaço de endereçamento virtual de 64Kb (16 bits). O tamanho da memória principal do computador é de 32Kb, e o tamanho de uma unidade é de 4Kb. Com isso, temos 16 páginas virtuais e 8 molduras de página. O número dentro de uma página virtual indica a moldura de página na qual esta página está mapeada, sendo que o 'X' indica que a página não está mapeada na memória.



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

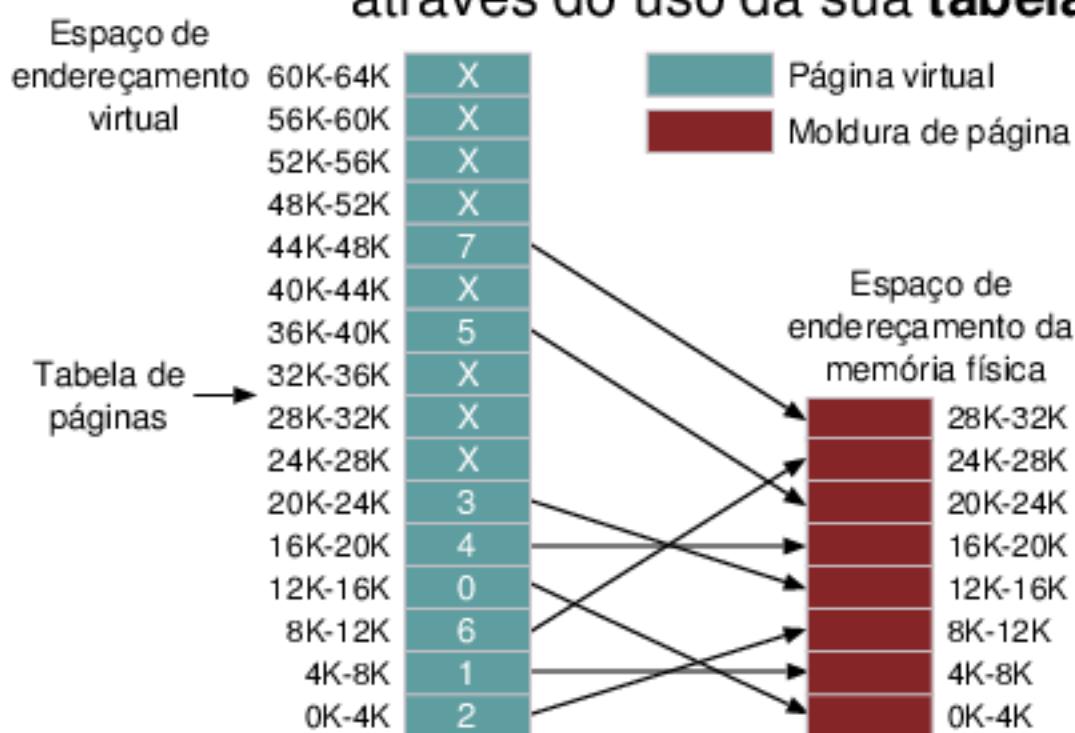


Se o programa executa uma instrução que acessa o endereço virtual 0, como, por exemplo, a instrução MOV REG,0, então a MMU, ao receber o endereço do processador, descobrirá que este está na página virtual 0 (endereços de 0 até 4095). Como esta página está mapeada na moldura de página 2 (endereços de 8192 até 12287), o endereço físico que a MMU irá gerar será o 8192.



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

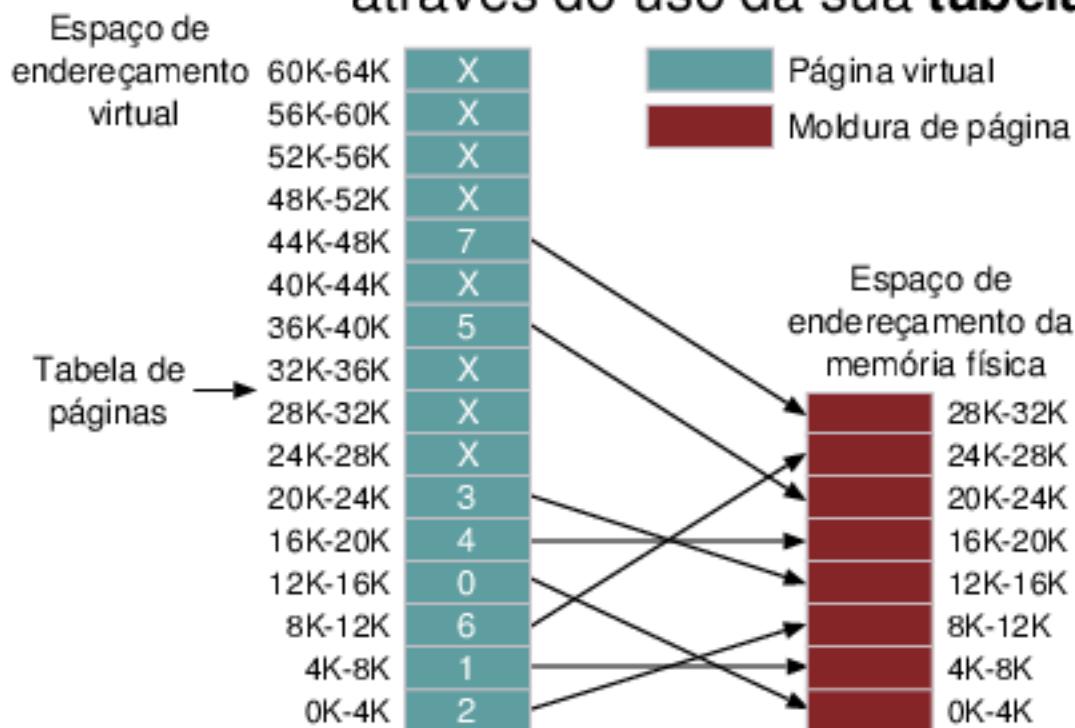


Se agora o programa executar uma instrução MOV REG, 20500, que acessa o endereço virtual 20500, então o endereço físico gerado pela MMU será agora o 12308, pois o endereço 20500 está na página virtual 5 (endereços de 20480 até 24575), que é mapeada para a moldura de página 3 (cujos endereços vão de 12288 à 16383), e está a 20 bytes do início desta página.



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

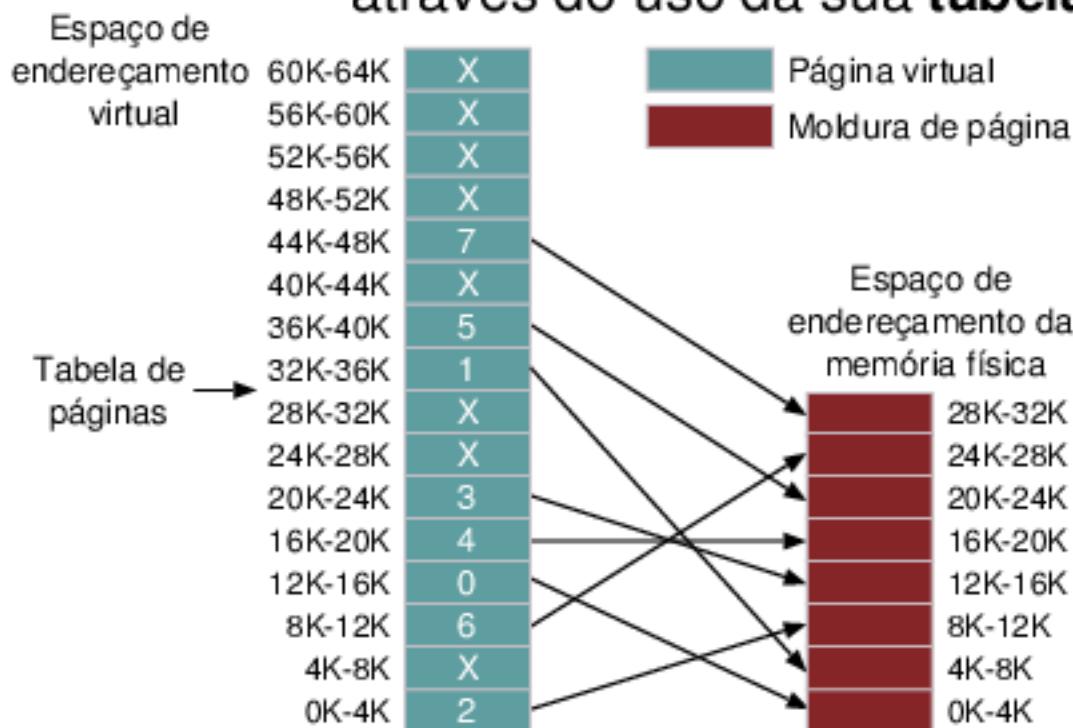


Suponha que o programa executou uma instrução que acessa o endereço virtual 32780, que está dentro da página virtual 8. Como a página não está mapeada na memória (pois temos um 'X' na entrada), a MMU gera uma interrupção, chamada de **falha de página**, para que o sistema possa carregar a página acessada na memória. Em geral, o hardware possui um bit ausente/presente para indicar se a página está ou não mapeada.



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

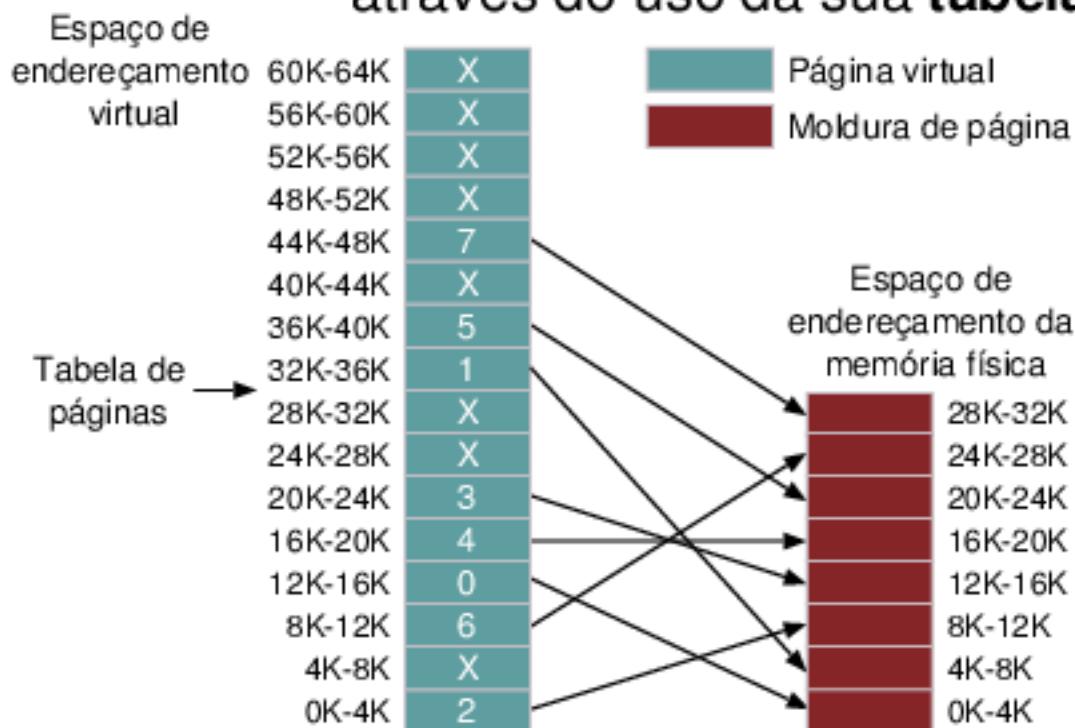


Suponha que o sistema decidiu carregar a página virtual 8 na moldura de página 1, salvando o conteúdo da página virtual 1, que estava mapeada nesta moldura, no disco, se necessário. Então, além de copiar a página virtual 8 para a moldura de página 1, o sistema deverá atualizar a entrada 1 da tabela de páginas para X (pois a página 1 agora não está mais mapeada), e a entrada 8 da tabela para a moldura de página 1.



Memória virtual

- Se usamos a técnica de **paginação** ao gerenciar a memória virtual:
- O endereço virtual é dividido em um conjunto de unidades que são chamadas de **páginas virtuais**.
 - A memória principal do computador também é dividida nestas unidades, que são chamadas de **molduras de página**.
 - A MMU mapeia as páginas virtuais em molduras de página, através do uso da sua **tabela de páginas**:

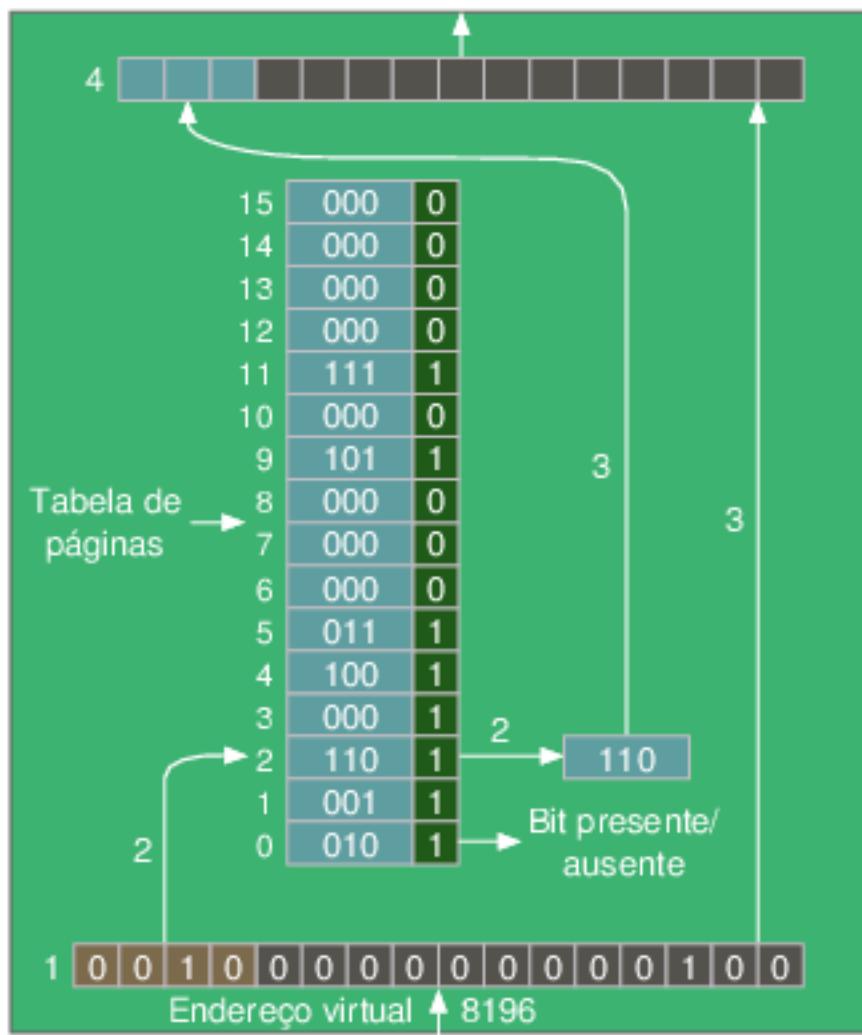


Depois de copiar a página virtual 8 para a moldura de página 1, a instrução que acessou o endereço 32780 é reiniciada, e este endereço é convertido para o endereço físico 4108, pois este está a 12 bytes do endereço inicial da página (que é 32768), e o primeiro endereço da moldura de página 1 é o 4096.



Memória virtual

→ A conversão de um endereço virtual para um endereço físico pela MMU é feita do seguinte modo:



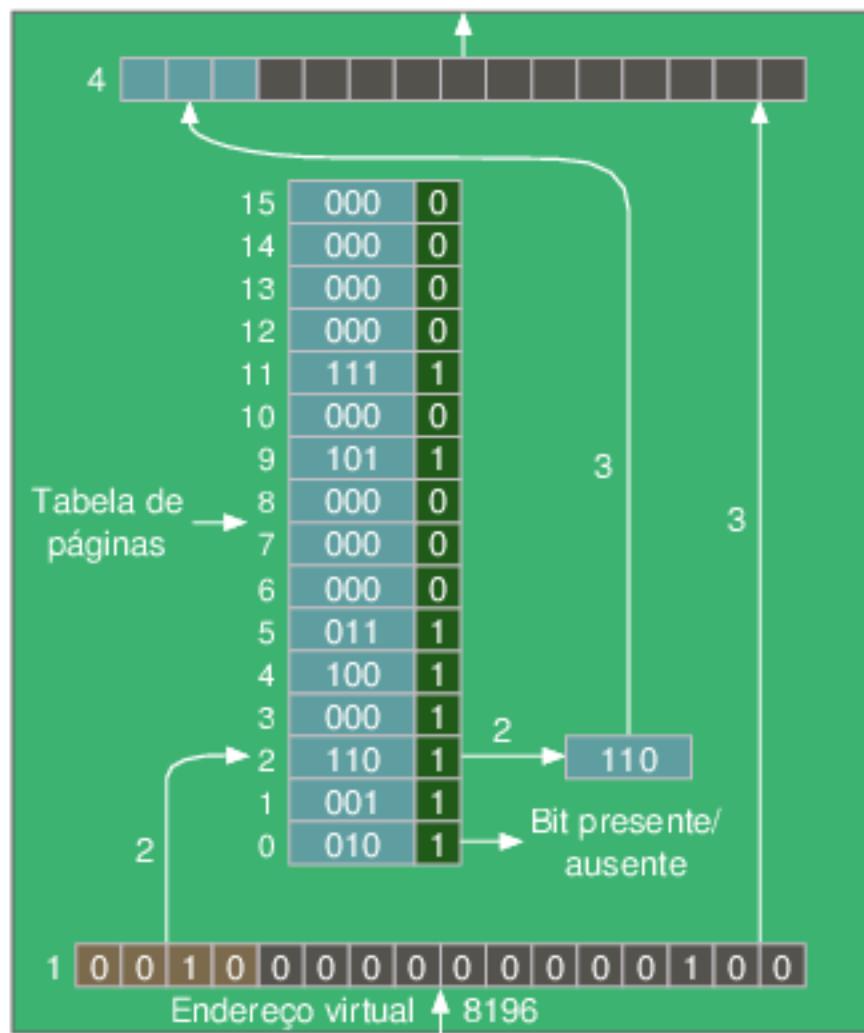
Funcionamento interno da MMU. O endereço virtual é dividido em dois campos: o **número da página** e o **deslocamento** dentro desta página. O número da página é usado como um índice na tabela de páginas para obter o número da moldura de página com esta página. O endereço físico enviado pela MMU à memória será composto por este número da moldura, e pelo deslocamento dentro desta página, que será o mesmo dentro da moldura.

Vamos ver como a MMU funciona mais de perto, vendo como o endereço virtual 8196 é convertido no endereço físico 24580, num computador com memória física de 32K, com 64K de espaço de endereçamento virtual, e com páginas com tamanho de 4K.



Memória virtual

→ A conversão de um endereço virtual para um endereço físico pela MMU é feita do seguinte modo:

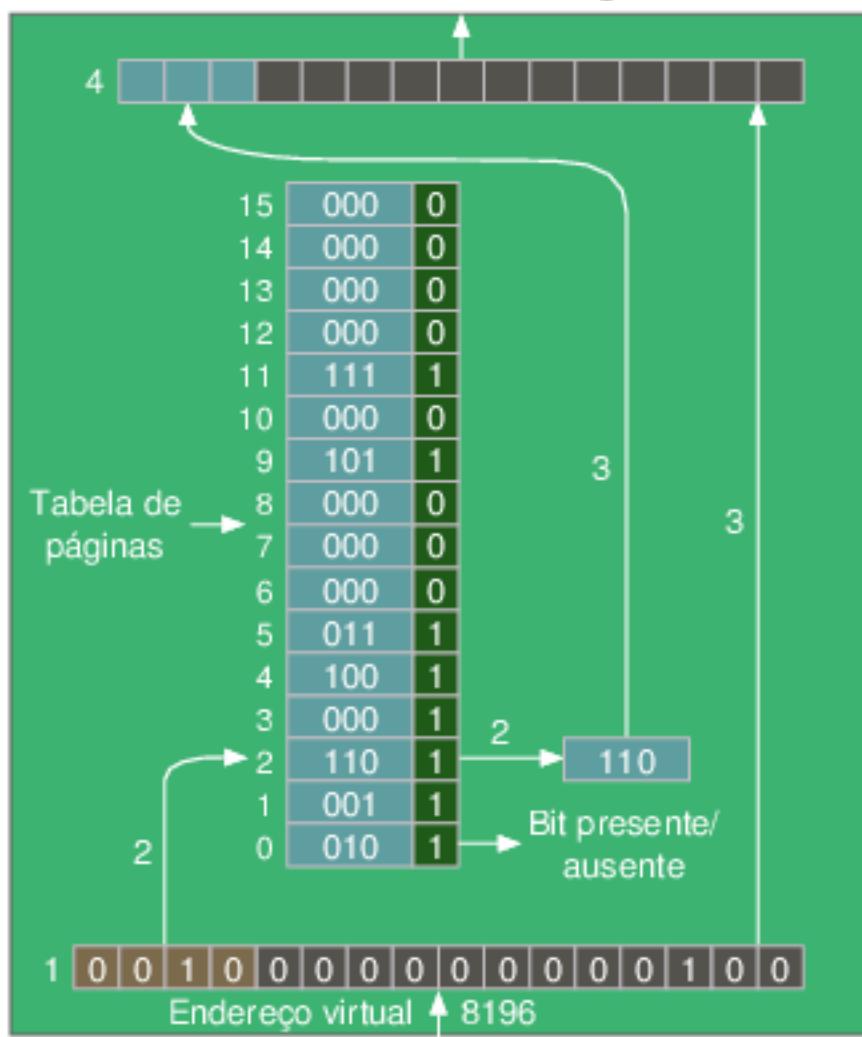


Passo 1: A MMU recebe o endereço virtual 8196 do processador. Como o espaço de endereçamento virtual é de 64K, e como o tamanho das páginas é de 4K, o número de bits no endereço virtual será de 16, sendo que os 4 bits superiores definirão a página, e os 12 bits inferiores darão o descolamento dentro desta página. No nosso exemplo, o endereço está a 4 bytes do início da página 2 (que começa no endereço 8192).



Memória virtual

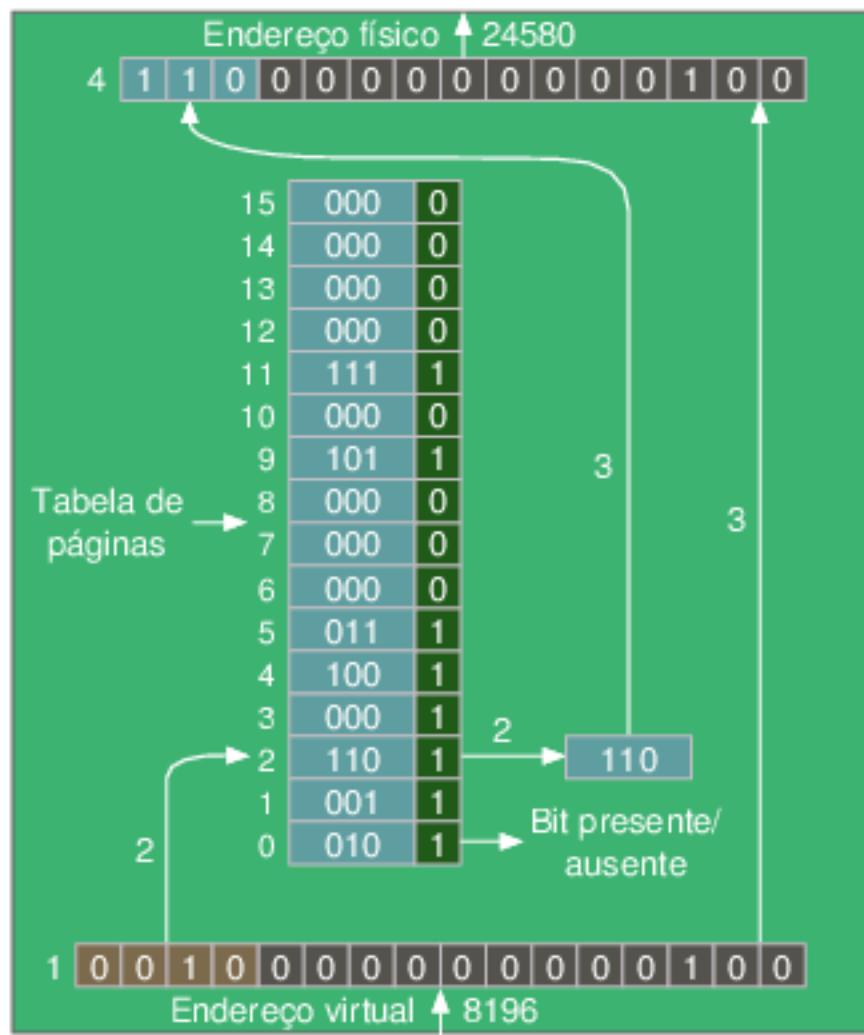
→ A conversão de um endereço virtual para um endereço físico pela MMU é feita do seguinte modo:



Passo 2: o número da página, 2, é usado como um índice na tabela de páginas para obter a entrada desta página. Como o bit ausente/presente da entrada é igual a 1, o valor 6 (110 em binário) da entrada indica que esta página foi copiada para a moldura de página 6. Se o bit ausente/presente fosse igual a 0, então a MMU iria gerar uma falha de página, para que o sistema operacional possa escolher uma moldura em que colocar a página que acabou de ser referenciada.

Memória virtual

→ A conversão de um endereço virtual para um endereço físico pela MMU é feita do seguinte modo:

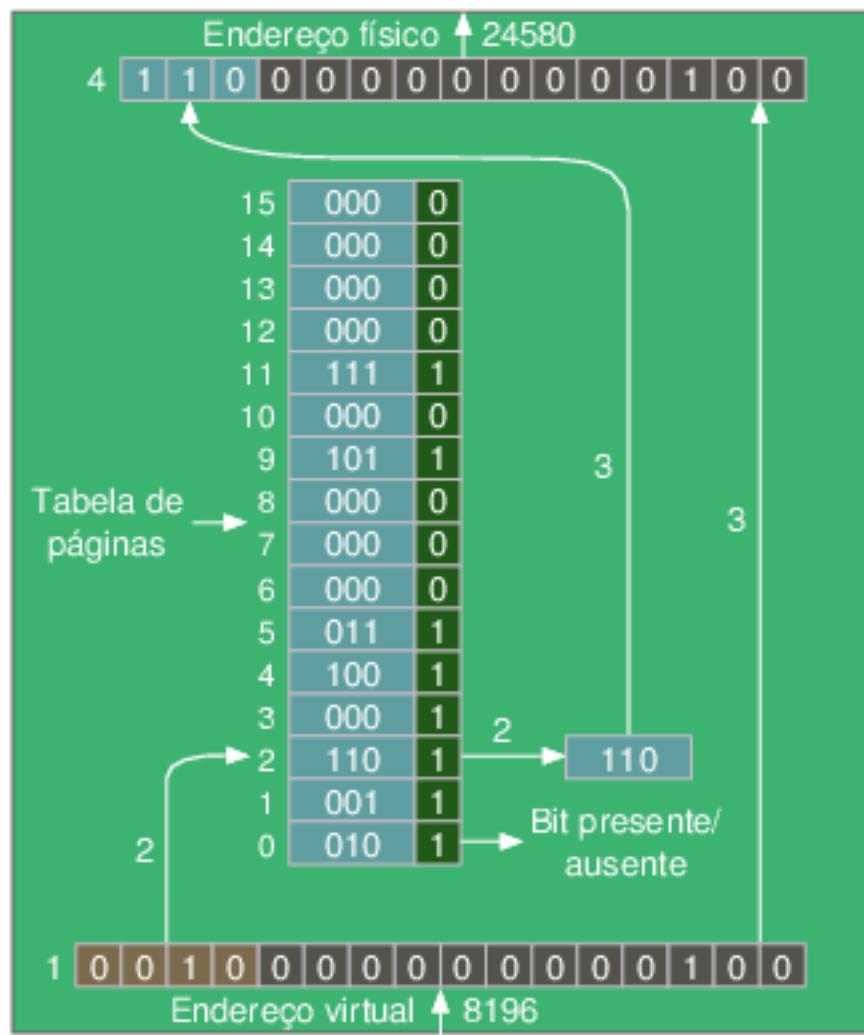


Passo 3: o endereço físico que a MMU vai enviar para a memória é formado, sendo que este será composto pelo deslocamento de 12 bits nos seus bits inferiores, e por 3 bits superiores, que indicam a moldura de página que possui uma cópia da página, pois o tamanho da memória é de 32K, e o tamanho da moldura é de 4K. No nosso exemplo, o valor 6 lido da tabela irá compor o endereço físico, juntamente com o deslocamento de 4, o que formará o endereço físico 24580, que está a 4 bytes do início da moldura de página 6 (que começa no endereço 24576).



Memória virtual

→ A conversão de um endereço virtual para um endereço físico pela MMU é feita do seguinte modo:



Passo 4: o endereço físico 24580 é colocado no barramento, para que a instrução que acessou o endereço virtual 8196 possa ser corretamente executada.

