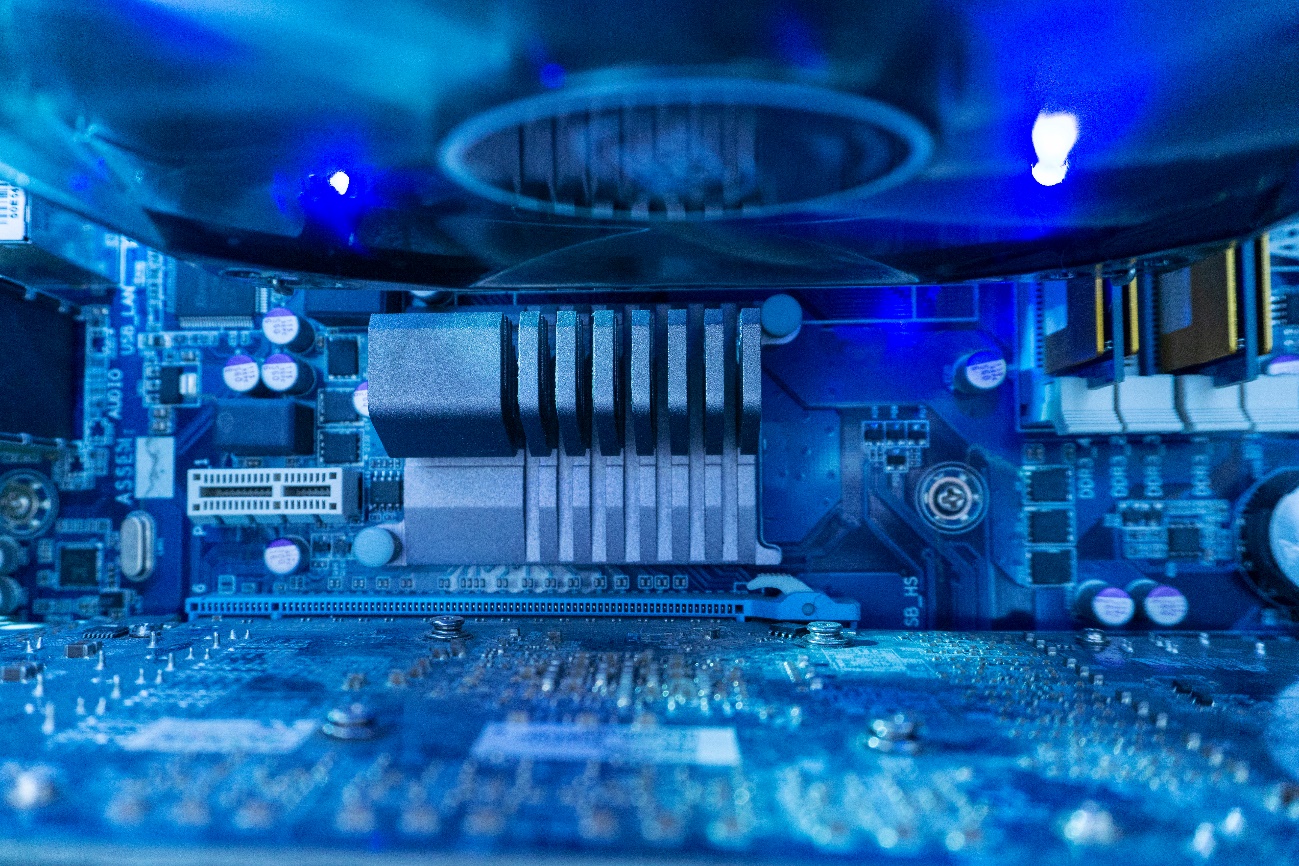
**Introdução da aula**



**Qual é o foco da aula?**

Nesta aula, vamos apresentar os conceitos de memória virtual e suas características, como se dá a paginação, a tabela de páginas e a segmentação, além de entender o funcionamento dos algoritmos de substituição de páginas.

**Objetivos gerais de aprendizagem**

Ao longo desta aula, você irá:

* descrever os conceitos e as características da técnica memória virtual;
* explicar os objetivos e o funcionamento dos algoritmos de substituição de páginas;
* analisar os espaços de endereçamento independentes chamados de segmentos.

**Situação-problema**

A memória virtual é um espaço reservado no disco rígido do computador para ser utilizado quando a memória RAM não é suficiente para executar os processos. Neste caso, uma parte dos processos ficam em disco aguardando a liberação da memória para executar enquanto outros processos estão em execução.

Para entendermos melhor o conceito de memória virtual, observe o seguinte exemplo: suponha que você quer executar um aplicativo de restauração de imagens que possui um tamanho de 100 MB e um programa de edição de textos com um tamanho de 60 MB, porém a memória RAM do seu computador tem espaço para comportar programas de até 145 MB.

Neste caso, seria carregado na memória RAM o aplicativo de restauração de imagens (com 100 MB) e uma parte do programa de edição de textos (45 MB) e o restante ficaria em disco para serem executados posteriormente a medida que a memória fosse liberada.

Relembrando nosso contexto, você trabalha em uma empresa de infraestrutura como gerente de T.I. e irá representá-la em uma importante feira de tecnologia. Durante a feira, um proprietário de uma loja de jogos digitais está à procura de ajuda para resolver o problema de baixa memória virtual dos computadores de sua loja e procura o seu *stand*.

Ele alega que quando os computadores executam vários programas de uma vez e o consumo de memória é mais elevado, os mesmos chegam no limite do consumo e travam, além de fechar os programas que estão abertos. Assim surgem os seguintes questionamentos:

* o que aconteceu no computador neste caso?
* qual a solução que você dará para este problema?
* quais são os algoritmos de substituição de páginas mais eficientes em relação ao desempenho do sistema?

Para que você consiga responder a esses e a outros questionamentos sobre memória virtual, vamos conhecer mais sobre eles, e, assim, você conseguirá entregar um relatório para o seu gestor, com os anteprojetos elaborados durante a feira.

Bons estudos!

**Técnicas de gerência de memória**



Segundo Tanenbaum (2003), um dos grandes desafios dos programadores era lidar com programas maiores que a memória física disponível. Neste caso, a técnica de *overlay* (sobreposição) era utilizada para dividir o programa em módulos em que cada um destes era carregado em memória ou permanecia em disco à medida que eram solicitados.

Porém, a divisão do programa em *overlay* era feita pelo programador, o que ocasionava demora em seu trabalho. Com isso, foi desenvolvida uma técnica cuja divisão do programa era feita pelo sistema operacional, chamada memória virtual.

**Memória Virtual**

Segundo Tanenbaum (2003), um computador que utiliza memória virtual permite que o volume de informações de um programa como código dado e pilha ultrapasse a quantidade total de memória física disponível para ele, mantendo as partes ativas na memória e as demais no disco rígido. A memória virtual é um arquivo dinâmico e de tamanho variável na maioria dos sistemas operacionais.

Segundo Machado e Maia (2007), a memória virtual permite que vários processos compartilhem a memória principal, uma vez que, somente algumas partes dos processos estarão ativas na mesma, possibilitando uma utilização eficiente do processador e reduzindo a fragmentação da memória principal. Em sistemas com memória virtual é utilizada uma técnica chamada paginação que será descrita a seguir.

\_\_\_\_\_\_

**➕ Pesquise mais**

A utilização da memória virtual permite o compartilhamento da memória física entre vários aplicativos ao mesmo tempo. Além disso, cada programa enxerga uma versão virtual própria da memória.

Para saber mais sobre esse assunto indicamos o artigo [**Como funciona a memória virtual?**](https://www.tecmundo.com.br/internet/2190-como-funciona-a-memoria-virtual-.htm), do autor Gabriel Gugik.

\_\_\_\_\_\_

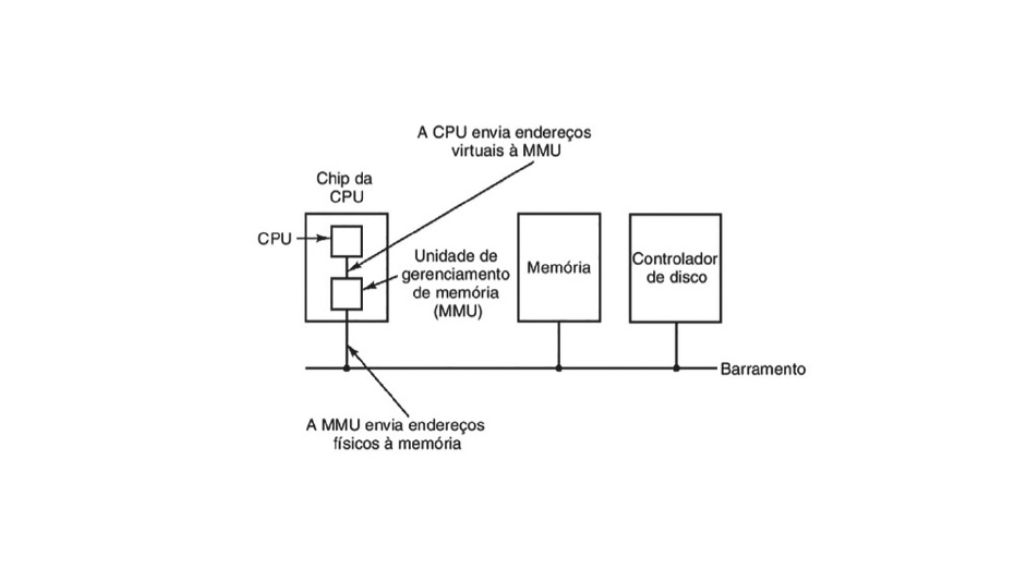
**Paginação e tabela de páginas**

Segundo Machado e Maia (2007), a paginação é a técnica de gerência de memória em que o endereçamento virtual e o espaço de endereçamento real são divididos em blocos do mesmo tamanho, chamado páginas. Foi criada para fornecer um espaço de endereçamento linear sem a necessidade de adquirir mais memória física.

Segundo Tanenbaum (2003), os programas geram endereços virtuais e constituem o espaço de endereçamento virtual. Nos sistemas operacionais que não possuem memória virtual, o endereço virtual e o endereço físico são os mesmos.

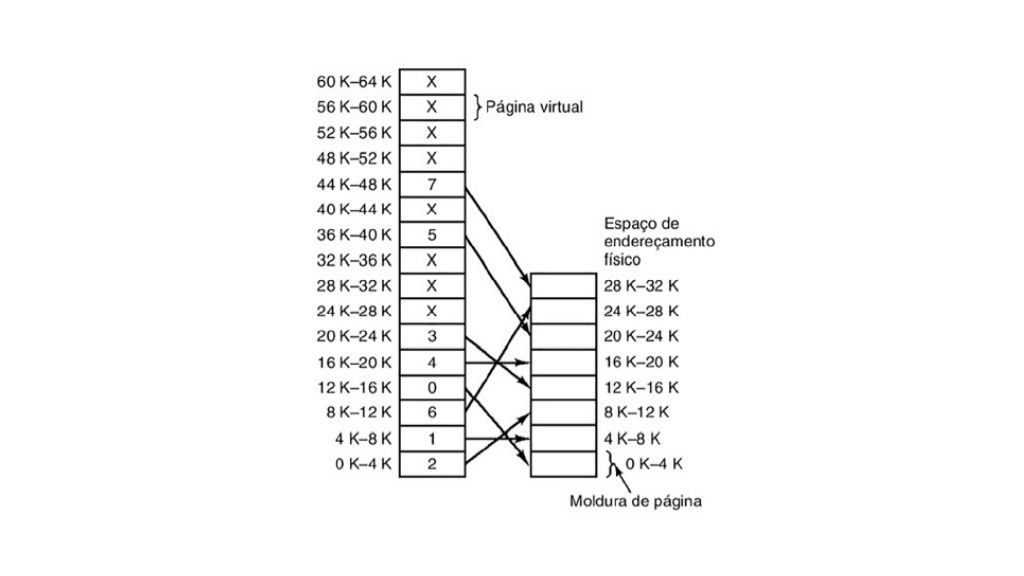
Já nos sistemas operacionais que trabalham com a memória virtual, o endereço virtual é enviado para a MMU (*memory management unit*, em que um chip está localizado na CPU (*Central Processing Unit* ou unidade central de processamento) que mapeia (estrutura) os endereços virtuais em endereços físicos, conforme apresentado na figura a seguir.

A CPU gera os endereços virtuais e os envia a MMU, que por sua vez envia os endereços físicos para a memória.

Localização da MMU e sua função. Fonte: Tanenbaum (2003, p. 150).

Ainda segundo Tanenbaum (2003), a figura “MMU – mapeamento de endereço virtual em físico” apresenta o funcionamento do mapeamento de endereços virtuais para endereços físicos realizado pela MMU. O computador possui uma memória virtual de 64 KB (gerando endereços virtuais de 16 bits de 0 a 64 K) e uma memória física de 32 KB.

Programas podem ser maiores que o tamanho da memória física, mas não podem ser totalmente carregados nela, devendo permanecer em disco. O endereço virtual divide-se em unidades conhecidas como páginas e sua referência na memória física são as molduras de página. As páginas e as molduras de páginas têm o mesmo tamanho e a movimentação entre disco e memória são sempre realizadas em unidades de página.

MMU – mapeamento de endereço virtual em físico. Fonte: Tanenbaum (2003, p. 151).

Na figura acima existem 16 páginas virtuais e oito molduras de página. Quando um programa tenta acessar o endereço 0, o endereço virtual 0 é enviado a MMU, que o localiza na página virtual (0 a 4 K) e que corresponde a moldura de página 2 (8 K a 12 K).

A MMU então transforma o endereço virtual 0 no endereço físico 8 K e o envia à memória por meio do barramento. Como existem apenas oito molduras de página física, somente oito páginas virtuais podem ser mapeadas.

Se um programa acessar uma página que não esteja mapeada (como a página entre 32 k e 36 k), a MMU faz esta validação e desvia para o sistema operacional. Esta interrupção (*trap*) é chamada de falta de página.

Segundo Tanenbaum (2003), na falta de página o sistema operacional seleciona uma moldura de página para salvar no disco (por exemplo, a moldura 1), marca na tabela de páginas virtuais a entrada escolhida como não mapeada (marcando com um X), carrega a página virtual referenciada pela instrução na moldura de página recém liberada (substituindo o X pela moldura 1), e marca na tabela de páginas virtuais a entrada da nova página virtual (página 8) como mapeada (moldura 1).

Ou seja, a MMU transformará o endereço virtual 32 K no endereço físico 4 K. O mapeamento realizado pela MMU dá-se por meio da tabela de páginas. O objetivo da tabela é mapear páginas virtuais em molduras de página física.

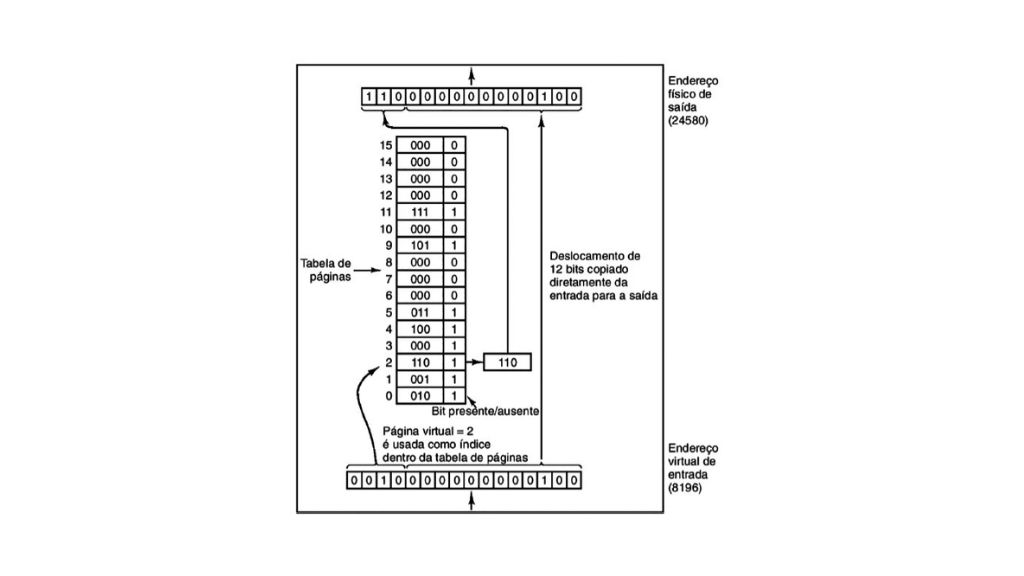
Segundo Silberschatz, Galvin e Gagne (2015), a tabela de páginas contém o endereço virtual de cada moldura de página na memória física, e o número da página é utilizado como um índice na tabela. Cada processo possui sua tabela própria e cada página possui uma entrada nela.

Segundo Tanenbaum (2003), cada página na tabela possui um bit presente/ausente. Se o bit for 0 (zero), indica uma interrupção por falta de página e caso o bit tenha o valor 1, a página está mapeada na memória. O número da moldura deve ser concatenado com os bits de deslocamento formando o endereço físico.

A figura “Operação interna da MMU com 16 páginas” apresenta um exemplo de como a MMU trabalha junto comatabela de páginas.

O endereço virtual, 8196 (0010000000000100), é mapeado para o endereço físico 24580 (endereço físico de saída). O endereço virtual de 16 bits (0010000000000100) que chega a MMU é dividido num número de página de 4 bits (0010) mais significativos e um deslocamento de 12 bits (000000000100) menos significativos. O número da página (0010 = 2) é usado como índice para a tabela de páginas.

Se o bit presente/ausente for 1, o número da moldura de página encontrado na tabela será copiado para os três bits (110) mais significativos concatenados ao deslocamento de 12 bits do endereço virtual que não sofreu alteração (110000000000100). O registrador envia para a memória o endereço físico via barramento, conforme apresentado na figura.

Operação interna da MMU com 16 páginas. Fonte: Tanenbaum (2003, p. 152).

**Algoritmos de substituição de páginas**



Segundo Machado e Maia (2007), um dos maiores desafios na gerência de memória virtual por paginação é decidir quais as páginas devem ser liberadas. Quando uma falta de página ocorre, o sistema operacional precisa escolher uma a ser removida da memória, a fim de liberar espaço para uma nova ser trazida. Quanto menor for o tempo gasto com as recargas de páginas, mais eficiente será o algoritmo.

Segundo Tanenbaum (2003), se na memória a página teve alteração, ela deverá retornar ao disco rígido e atualizar a cópia da página virtual que está lá. Senão, não é preciso retornar com a página para o disco. Nos casos em que ocorre falta de página, pode ser escolhido de forma aleatória uma página a ser removida, porém é melhor escolher uma página que não foi muito usada.

Segundo Machado e Maia (2007), os algoritmos de substituição de páginas têm o objetivo de selecionar as páginas com as menores chances de serem referenciadas (utilizadas) no futuro. A seguir veremos os algoritmos de substituição de páginas.

* **Algoritmo de substituição de página ótimo**

Segundo Machado e Maia (2007), o algoritmo de substituição de página ótimo seleciona uma página que não será referenciada no futuro ou aquela que demorará a ser utilizada novamente. Este algoritmo garante uma menor paginação, porém é impossível de ser implementado, uma vez que o sistema operacional não consegue prever o futuro das aplicações e saber quando cada página será referenciada novamente.

* **Algoritmo de substituição de página não recentemente utilizada (NUR)**

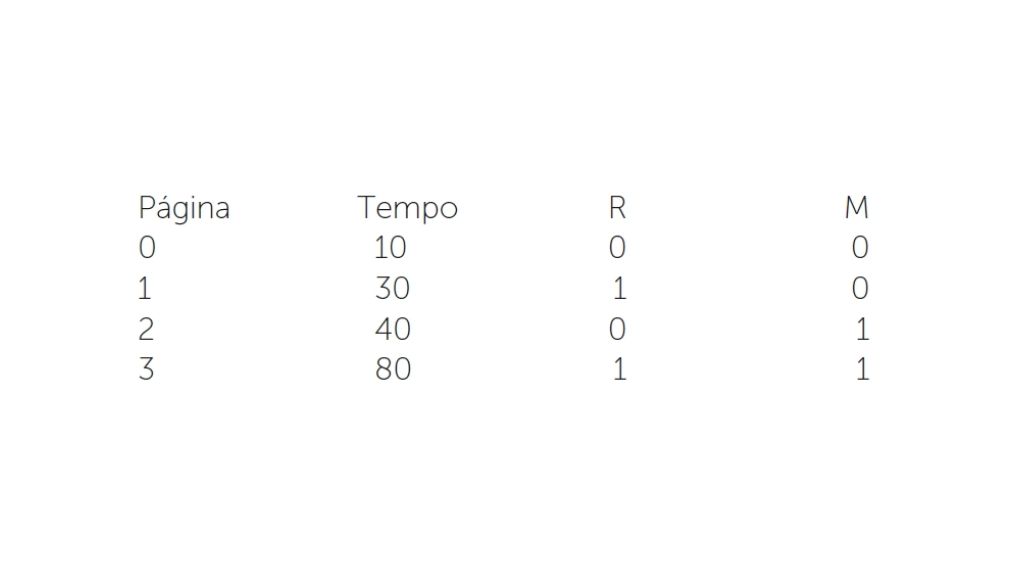
Segundo Tanenbaum (2003), muitos computadores que utilizam a memória virtual possuem dois bits R (referenciada) e M (modificada) associados a cada página virtual para identificar quais páginas físicas estão sendo usadas e quais não estão. No algoritmo NUR, quando um processo é iniciado os bits R e M são colocados em 0 para todas as páginas. Periodicamente, o bit R é limpo de modo que diferencie as páginas que não foram referenciadas recentemente daquelas que foram.

Quando acontece uma falta de página, o sistema operacional verifica todas as páginas e as separa em quatro categorias, com base nos bits R e M.

* Classe 0: não referenciada, não modificada.
* Classe 1: não referenciada, modificada.
* Classe 2: referenciada, não modificada.
* Classe 3: referenciada, modificada.

O algoritmo remove aleatoriamente uma página de classe de ordem mais baixa que não esteja vazia, ou seja, remove uma página modificada, mas não referenciada. O algoritmo NUR é simples e de fácil implementação.

Por exemplo, suponha que as páginas virtuais 0, 1, 2 e 3, iniciaram todas com os valores zero. Num determinado momento foram associados aos bits referenciada e modificada os seguintes valores:



Como o algoritmo retira a página de ordem mais baixa, que esteja vazia e que tenha sido modificada, mas não referenciada, as páginas 0 e 2 poderiam ser removidas segundo o algoritmo NUR.

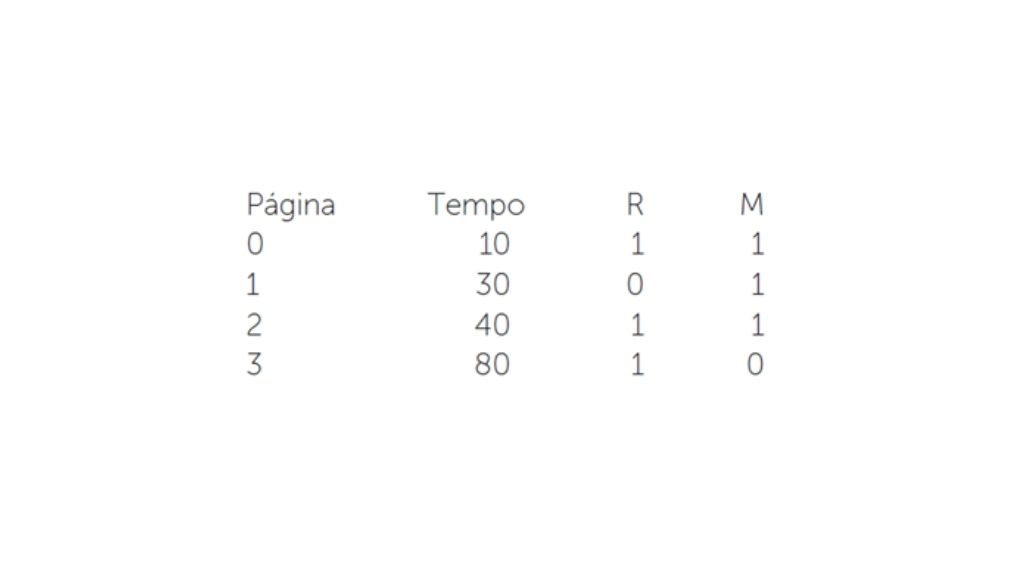
* **Algoritmo de substituição de página FIFO**

Segundo Machado e Maia (2007), no algoritmo de substituição de páginas FIFO (*first in first out*), a primeira página utilizada será a primeira a ser escolhida para ser removida. As páginas são inseridas em uma fila sendo que as mais antigas estão no início e as mais recentes no final. É um algoritmo de baixo custo, porém quase não é implementado em função de escolher páginas antigas que sempre são referenciadas.

\_\_\_\_\_\_

**📝 Exemplificando**

Para exemplificar o funcionamento do algoritmo FIFO, suponha que as páginas virtuais 0, 1, 2 e 3, iniciaram todas com os valores zero e estão em fila. A página 0 foi a primeira página a ser utilizada no tempo 10, logo em seguida a página 1 no tempo 30, depois a página 2 no tempo 40 e por último a página 3 no tempo 80.



Como o algoritmo FIFO desconsidera as páginas que foram referenciadas na última utilização, independentemente de sua importância, a página 0 seria removida.

\_\_\_\_\_\_

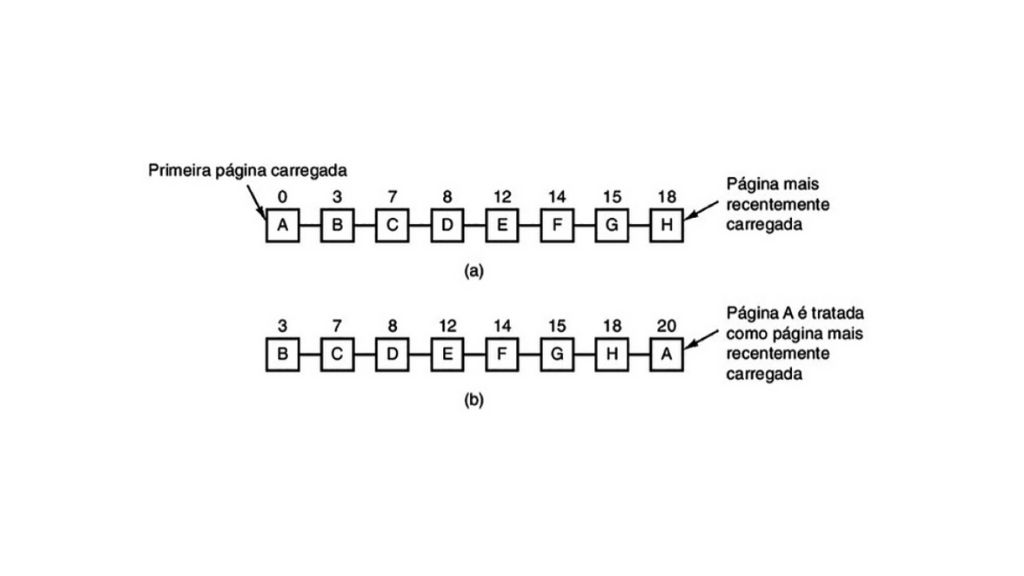
* **Algoritmo de substituição de página de segunda chance (SC)**

O algoritmo FIFO não leva em consideração a retirada de uma página usada constantemente. Segundo Tanenbaum (2003), uma alteração simples evitaria a eliminação de uma página muito usada. Basta verificar o bit de referência da página mais antiga.

Se o bit R for 0, a página não está sendo usada e pode ser substituída. Se o bit R for 1 será colocado em 0, e essa página será inserida no final da lista como se ela tivesse acabado de ser carregada na memória.

A figura “Algoritmo segunda chance”(a) apresenta uma lista com as páginas de A até H ordenadas por ordem de chegada na memória (a página A chegou no instante 0, a página B no instante 3, a página C no instante 7, e assim por diante).

Caso ocorra uma falta de página no instante 20, o bit R da página A (que é a mais antiga) é verificado. Se o valor do bit R for 0 (zero) a página é retirada da memória, caso contrário A será inserida no final da fila no instante 20 (figura (a)) e R receberá o valor 0. O algoritmo continua a percorrer a lista a partir da página B para encontrar uma página que possa ser substituída.

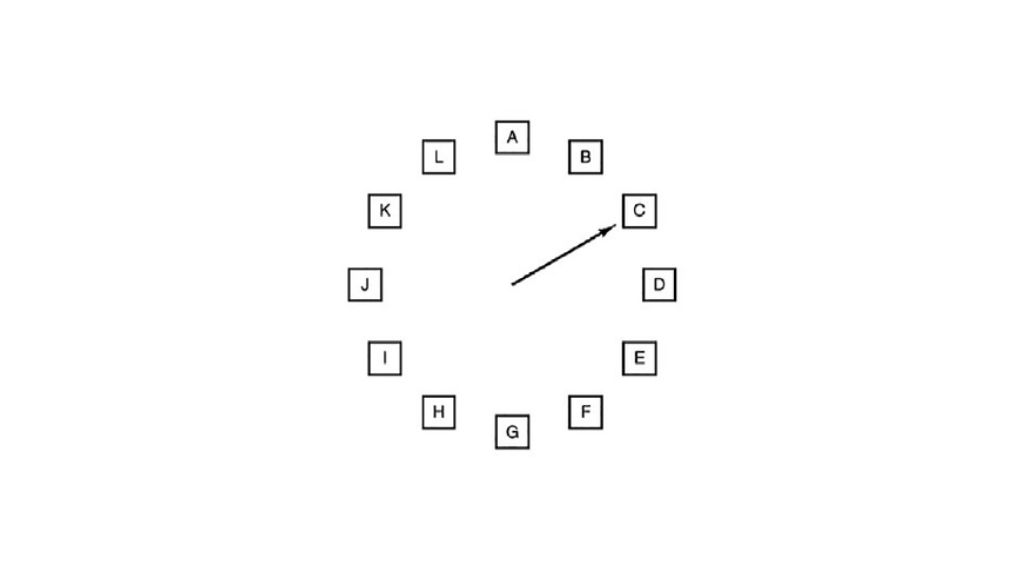
Algoritmo segunda chance. Fonte: Tanenbaum (2003, p. 162).

* **Algoritmo de substituição de página relógio**

Este algoritmo mantêm todas as páginas em uma lista circular em forma de relógio e um ponteiro apontando para a ‘cabeça’ da lista, ou seja, para a página mais antiga. A figura “Algoritmo de substituição de página relógio”apresenta o funcionamento do algoritmo, (TANENBAUM, 2003).

Quando ocorre uma falta de página, o ponteiro aponta para a página mais antiga e o bit R é verificado (página C). Se o valor do bit R for 0 (zero) a página é retirada da memória e o ponteiro avança uma posição. Caso contrário, R receberá o valor 0 e o ponteiro avançará para a página mais antiga. A diferença entre o algoritmo SC para o algoritmo relógio está na implementação.

O algoritmo SC é implementado através de fila e o relógio através de uma lista circular.

Algoritmo de substituição de página relógio. Fonte: Tanenbaum (2003, p. 162).

* **Algoritmo de substituição de página menos recentemente utilizada (MRU)**

Segundo Tanenbaum (2003), este algoritmo é baseado na observação de que páginas referenciadas intensamente nas últimas instruções provavelmente serão novamente utilizadas, e páginas que não foram referenciadas não serão utilizadas na próxima instrução.

Este algoritmo aproxima- se do desempenho do algoritmo ótimo e possui uma implementação onerosa, pois mantem uma lista encadeada na memória com as páginas mais utilizadas no início da lista e as menos utilizadas no final, sendo necessário a sua atualização a cada referência de memória.

**Segmentação**

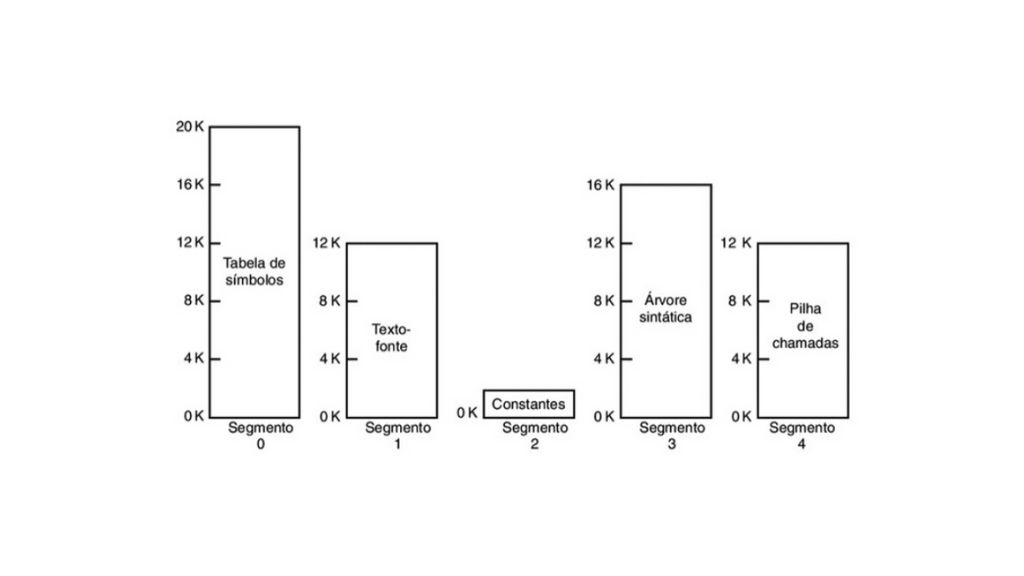


Segundo Tanenbaum (2003), a memória virtual é unidimensional, uma vez que inicia da posição 0 e vai até um endereço máximo. Em alguns casos isso pode ser um problema. Por exemplo, durante a execução de um compilador são gerados o texto fonte pré-processado, uma tabela de símbolos, uma tabela de constantes, uma árvore sintática e uma pilha.

O texto fonte, a tabela de símbolos, a tabela de constantes e a árvore sintática crescem continuamente durante a compilação, enquanto a pilha varia de tamanho. Se um programa possui um número grande de variáveis, o espaço reservado para elas na tabela de símbolos pode se esgotar à medida que o compilador é executado e sobrará espaço nas outras tabelas.

Para resolver este problema, basta fornecer ao computador vários espaços de endereçamento independentes chamados de segmentos. O segmento é uma unidade lógica que pode ser um vetor ou uma pilha, por exemplo, sendo de conhecimento do programador.

Cada segmento tem um tamanho dinâmico e independente dos outros (que varia de 0 a um valor máximo), conforme apresentado na figura “Memória segmentada”, permitindo que o segmento aumente ou diminua durante a execução. Os endereços são compostos pelo número do segmento e um deslocamento dentro do segmento.

Memória segmentada. Fonte: Tanenbaum (2003, p. 187).

A segmentação facilita o compartilhamento de procedimentos ou dados entre vários processos, por exemplo, o uso de uma biblioteca compartilhada, evitando que estas bibliotecas sejam replicadas em cada endereçamento de cada processo.

Segmentos podem ter diferentes proteções, como um segmento que armazena código pode ser definido apenas para execução, um segmento que armazena constantes, somente leitura e um segmento de pilha como leitura/escrita. As tentativas de acesso indevido podem ser capturadas e tratadas pelo *hardware* ou pelo sistema operacional.

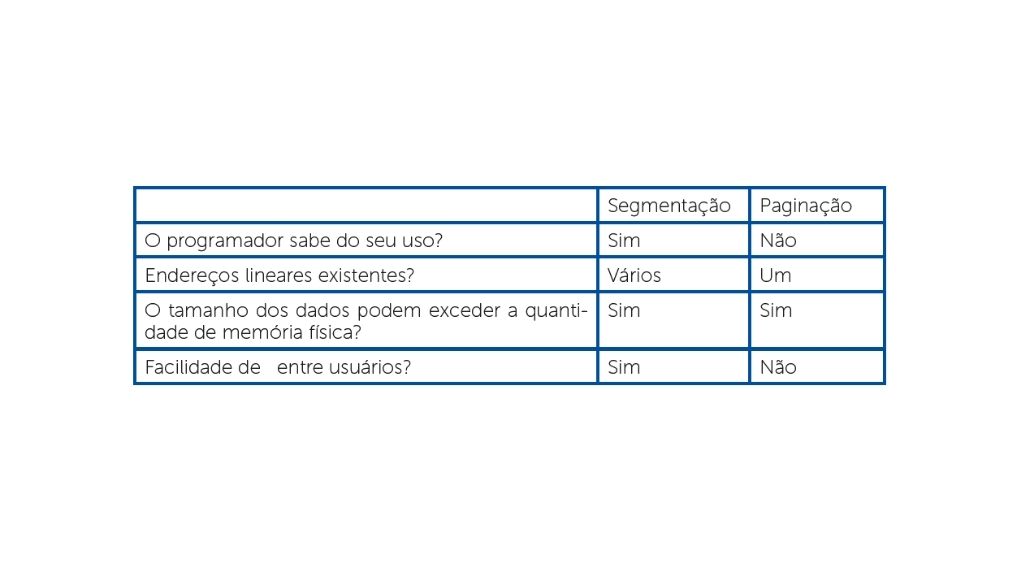
\_\_\_\_\_\_

**💭 Reflita**

Levando em consideração os computadores atuais em relação a quantidade de memória principal disponível, qual a melhor técnica a ser utilizada: paginação ou segmentação? Em relação à proteção qual a complexidade de implementação da segmentação? E da paginação?

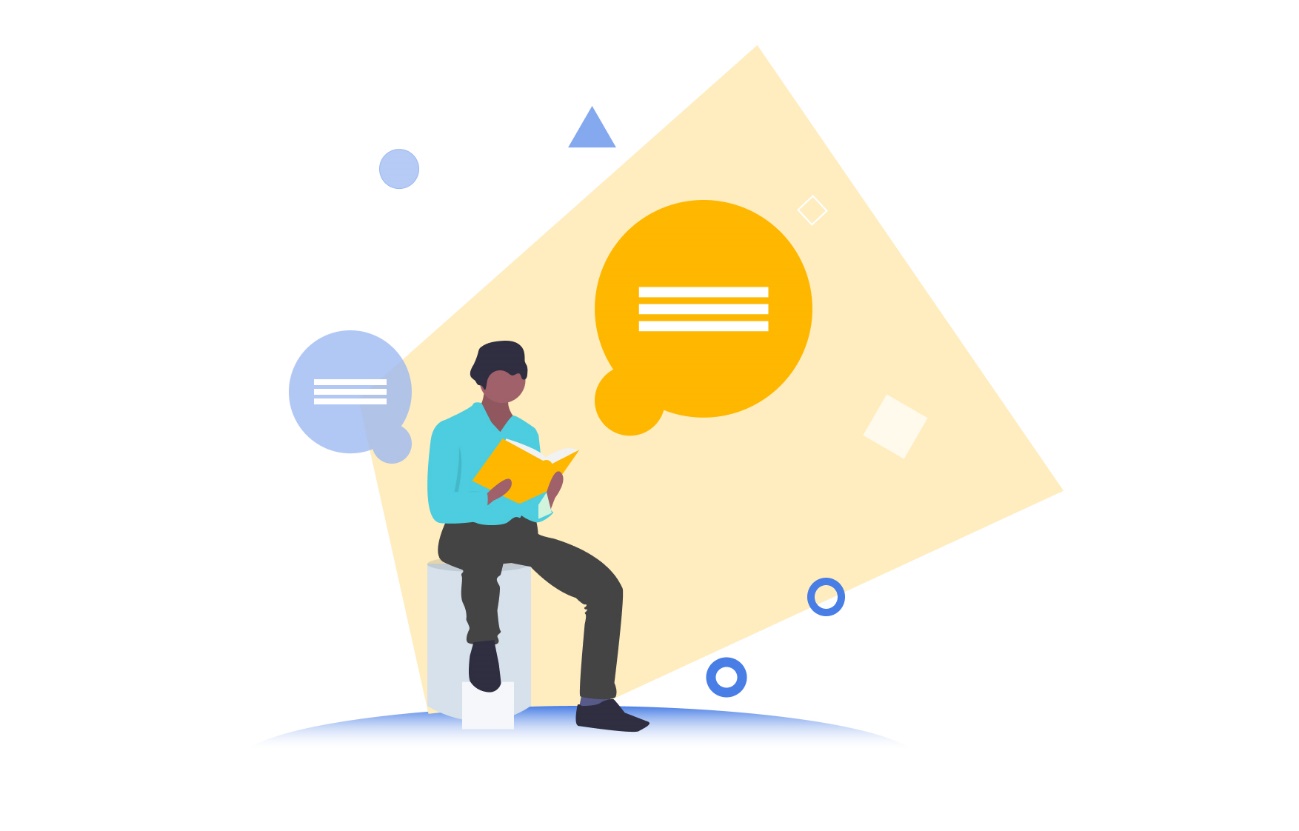
\_\_\_\_\_\_

Para finalizarmos, seguem as principais diferenças entre paginação e segmentação:

Critérios de segmentação e paginação. Fonte: elaborado pela autora (2018).

Pronto! Agora você já conhece os principais aspectos acerca da memória virtual! Encerramos aqui nossa disciplina e esperamos que a tenha aproveitado ao máximo! Em caso de dúvidas, não deixe de consultar os conteúdos já estudados!

**Conclusão**



Agora que você já aprendeu sobre os conceitos de memória virtual, paginação e segmentação, vamos relembrar o nosso contexto. Durante a feira, um proprietário de uma loja de jogos digitais está à procura de ajuda para resolver o problema de baixa memória virtual dos computadores de sua loja e procura o seu *stand*.

Diante dos problemas relatados por ele, surgem os seguintes questionamentos:

* o que aconteceu no computador neste caso?
* qual a solução que você dará para este problema?
* quais são os algoritmos de substituição de páginas mais eficientes em relação ao desempenho do sistema?

Neste caso, quando a mensagem de memória virtual baixa aparece é porque vários programas estão executando e consumindo memória, e o computador está no limite do seu uso, ou seja, pouca memória física disponível para processar o número de processos. Em alguns casos, ao iniciar o computador, nenhum processo consegue ser executado, e todos os programas abertos pelo usuário são fechados, uma vez que, dependendo da quantidade de memória e se existirem programas pesados sendo executados, isso irá acontecer.

A memória virtual, por ser um arquivo e funcionar como uma extensão da memória RAM, deve ser configurado de forma a atender a falta de memória física. O recomendado é que o arquivo de memória virtual tenha 1,5 o tamanho da memória física do computador. O tamanho do arquivo pode ser alterado manualmente para corrigir os problemas de travamento nos sistemas operacionais Linux e Windows.

Os algoritmos de substituição de páginas têm o objetivo de selecionar as páginas com as menores chances de serem referenciadas (utilizadas) no futuro, e escolher uma página a ser removida da memória a fim de liberar espaço para uma nova a ser trazida. Quanto menor for o tempo gasto com as recargas de páginas, mais eficiente será o algoritmo.

A escolha do melhor algoritmo a ser utilizado dependerá dos pré-requisitos definidos pelo *kernel* do sistema operacional. O algoritmo ótimo possui um bom desempenho, porém não é implementável. O algoritmo NUR é simples e de fácil implementação. Já o algoritmo FIFO não leva em consideração páginas que são constantemente utilizadas. O algoritmo SC é o algoritmo FIFO melhorado. O algoritmo relógio é realista e o algoritmo MRU é um bom algoritmo, porém difícil de ser implementado.

O relatório com os anteprojetos elaborados durante a feira e entregue para o gestor levou em consideração a necessidade de cada cliente e a solução proposta para eles. Para o cliente que possui pouca memória física em seu computador e não conseguem processar os dados da empresa foi sugerido ao cliente a aquisição de mais memória RAM para suportar o volume de dados.

Mesmo aumentando a quantidade de memória da empresa se as máquinas não conseguirem processar os dados corretamente, será necessário trocar o computador para um mais potente. Em relação ao estudante de tecnologia que questionou sobre o uso do *swap* que deixava a máquina lenta, foi sugerido que o *swapping* não pode ser utilizado como solução para a questão da falta de memória RAM.

Assim ele deve adquirir mais memória e reduzir a área de *swap* para melhor o desempenho do computador. Em relação ao cliente com a mensagem de memória virtual baixa o mesmo foi orientado a aumentar o tamanho do arquivo de memória virtual para auxiliar a execução dos programas.

**Referências**



BARTLETT, D. **Por dentro do gerenciamento de memória**. Disponível em: <https:// www.ibm.com/developerworks/br/library/l-memory/l-memory-pdf.pdf/>. Acesso em: 14 jun. 2018.

DAQUINO, F. **Adicionar memória RAM sempre vai deixar o meu PC mais rápido?** Disponível em: <https://www.tecmundo.com.br/memoria-ram/48168-adicionar-memoria-ram-sempre-vai-deixar-o-meu-pc-mais-rapido-.htm>. Acesso em: 04 jul. 2021.

FILHO, R. G. N. **Hierarquia de memórias**. Disponível em: <http://www.di.ufpb.br/ raimundo/Hierarquia/Hierarquia.html>. Acesso em: 11 jun. 2018.

GUGIK, G. **Como funciona a memória virtual?** Disponível em: <https://www.tecmundo.com.br/internet/2190-como-funciona-a-memoria-virtual-.htm>. Acesso em: 04 jul. 2021.

JUNIOR, L. A. P. L. **Gerência de memória**. Disponível em: <http://www.ppgia.pucpr. br/~laplima/ensino/so/materia/03\_memoria.html/>. Acesso em: 12 jun. 2018.

KLEINA, N. **Como funcionam as diferentes memórias quando o computador está em uso**. Disponível em: <https://www.tecmundo.com.br/armazenamento/9415-como-funcionam-as-diferentes-memorias-quando-o-computador-esta-em-uso.htm>. Acesso em: 04 jul. 2021.

MACHADO, F. B.; MAIA, L. P. **Arquitetura de sistemas operacionais**. 4. ed. Rio de Janeiro: LTC, 2007.

MORIMOTO, C. E. **Hardware para iniciantes, parte 3**. Disponível em: <https://www.hardware.com.br/tutoriais/hardware-iniciantes-3/pagina3.html>. Acesso em: 04 jul. 2021.

SALES E. **Memória virtual**. Disponível em: <https://www.oficinadanet.com.br/ artigo/730/memoria\_virtual>. Acesso em: 04 jul. 2021.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Fundamentos de sistemas operacionais**. 9. ed. Rio de Janeiro: LTC, 2015.

SIMIONI, D. **O que é a memória SWAP?** Disponível em: <https://www.diolinux.com.br/2014/09/o-que-e-memoria-swap.html>. Acesso em: 04 jul. 2021.

SOUZA, P. S. L.; BRUSCHI, S. M. **Por que estudar a hierarquia de memória?** Disponível em: <http://amnesia.lasdpc.icmc.usp.br/hierarquia-de-memoria/>. Acesso em: 04 jul. 2021.

TANENBAUM, A. S. **Sistemas operacionais modernos**. 2. ed. São Paulo: Pearson, 2003.