SISTEMAS OPERACIONAIS

AULA 14 – GERENCIAMENTO DE MEMÓRIA
MEMÓRIA VIRTUAL – PARTE 3

Prof.ª Sandra Cossul, Ma.

RELEMBRANDO...

- Todas as referências a um processo são **endereços lógicos** que são **dinamicamente traduzidos** em **endereços físicos** (reais) em **tempo de execução.**

- Um processo pode ser **trazido** ou **retirado** da **memória principal**, de forma que pode ocupar diferentes regiões da memória durante a sua execução.

- Um processo pode ser "quebrado" em pedaços (**páginas ou segmentos**) e esses pedaços precisam estar na memória principal para executar

- A tradução dos endereços é feita utilizado uma **tabela de páginas** ou **tabela de segmentos.**

INTRODUÇÃO

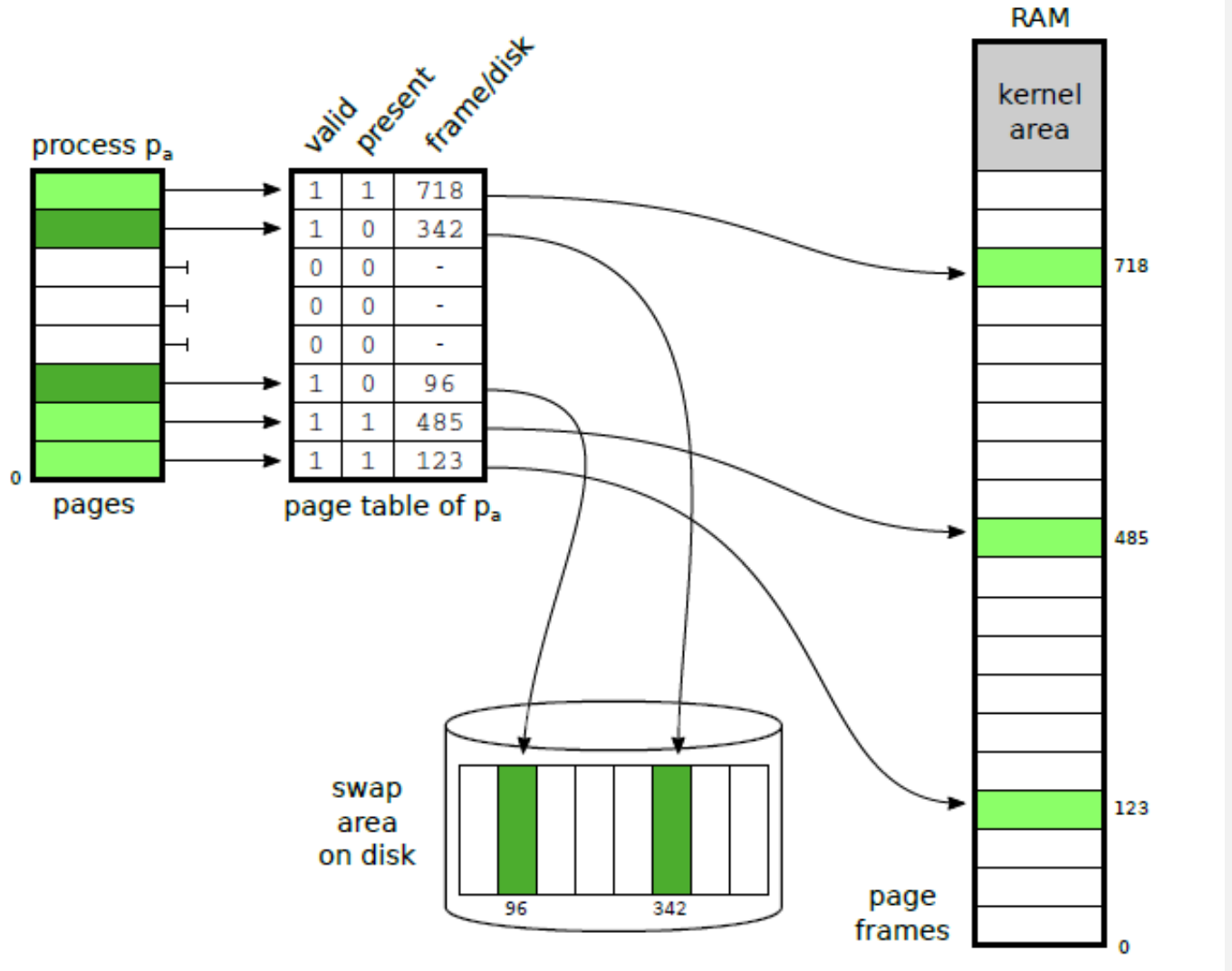
- **É preciso que todas as páginas ou todos os segmentos de um processo estejam na memória ao mesmo tempo?**
  - Não é necessário!
  - Se a página ou o segmento que contém a próxima instrução estiver na memória principal é o suficiente.

INTRODUÇÃO

- **Memória Virtual**
  - **Estratégia de alocação de memória em que a memória secundária pode ser endereçada como se fizesse parte da memória principal.**

INTRODUÇÃO

- **Memória Virtual**



INTRODUÇÃO

- **Como essa ideia pode ser implementada?**

- Um novo processo é trazido para a memória principal
- O SO inicia trazendo apenas um pedaço, que inclui o programa e os dados iniciais
- **Conjunto residente –** pedaço do processo presente na memória principal
- O processo executa e se o processador necessita um pedaço do processo que não está na memória, ele gera uma **interrupção,** indicado uma **falha de acesso à memória.**
- **SO** coloca este processo em um estado bloqueado
- Para este processo voltar a executar, o SO precisa trazer o pedaço do processo requerido pelo processador (requisição de leitura de disco)
- SO pode executar outro processo enquanto espera.
- Quando o pedaço do processo for trazido para a memória, o SO coloca ele na fila de pronto para executar.

INTRODUÇÃO

- **Quais as vantagens desta implementação?**

  - **Mais processos podem ser mantidos na memória principal**
    - como são carregados apenas partes de um processo, há espaço para mais "pedaços" de processos ao mesmo tempo.
    - mais eficiência no uso do processador

  - **Um processo pode ser maior que a memória principal**
    - processo não precisa ser carregado por inteiro, apenas os "pedaços"

MEMÓRIA VIRTUAL - INTRODUÇÃO

- **Memória real** → memória principal (RAM, limitada)
- **Memória virtual** → memória secundária (HD, SSD, muito mais espaço)

- **Memória virtual permite implementar a multiprogramação de forma efetiva, liberando o usuário da "preocupação" de limitação da memória principal.**

- Paginação e segmentação são utilizadas em conjunto com memória virtual!

RESUMO EXPLICATIVO

Paginação Simples	Paginação com Memória Virtual	Segmentação Simples	Segmentação com Memória Virtual
Memória principal **particionada** em pequenos blocos de mesmo tamanho chamados de **frames**		Memória principal **não particionada**	
Programa quebrado em páginas pelo compilador ou pelo gerenciador de memória		Programa quebrado em segmentos especificados pelo programador para o compilador (decisão do programador)	
Fragmentação interna (dentro dos frames)		Sem fragmentação interna	
Sem fragmentação externa		Fragmentação externa	
SO precisa manter uma tabela de páginas para cada processo anotando qual frame cada página ocupa		SO precisa manter uma tabela de segmentos para cada processo anotando a partir de qual endereço o segmento foi alocado e qual o seu tamanho	
SO precisa manter uma lista dos frames que estão livres na memória		SO precisa manter uma lista dos espaços livres da memória	

RESUMO EXPLICATIVO

Paginação Simples	Paginação com Memória Virtual	Segmentação Simples	Segmentação com Memória Virtual
CPU utiliza número da página e offset para calcular o endereço físico (absoluto)		CPU utiliza número do segmento e offset para calcular o endereço físico (absoluto)	
Todas as páginas de um processo devem estar na memória principal para ele executar.	Nem todas as páginas de um processo devem estar na memória principal para ele executar. Páginas podem ser trazidas conforme necessário.	Todas os segmentos de um processo devem estar na memória principal para ele executar.	Nem todas os segmentos de um processo devem estar na memória principal para ele executar. Segmentos podem ser trazidas conforme necessário.
	Ler uma página para a MP pode requerer a escrita de uma página em disco.		Ler um segmento para MP pode requerer a escrita de um ou + segmentos em disco.

MEMÓRIA VIRTUAL - EXEMPLO

- Considere um programa grande e um número de vetores de dados
- Em um período de tempo curto, a execução vai estar em uma pequena porção do processo e acessar somente uma parte dos dados
- Então, alocar o processo inteiro é desperdício de memória se apenas partes do processo serão executadas por vez.
- **Podemos fazer melhor uso da memória carregando apenas algumas partes do processo!**
- Caso o programa referencie uma parte do código ou precise de algum dado que não está na memória, é gerada uma **falha** e o SO traz o pedaço requisitado!

MEMÓRIA VIRTUAL - THRASHING

- O SO coloca e tira pedaços do processo da MP
- Quando ele coloca um novo pedaço, precisa retirar outro para abrir espaço

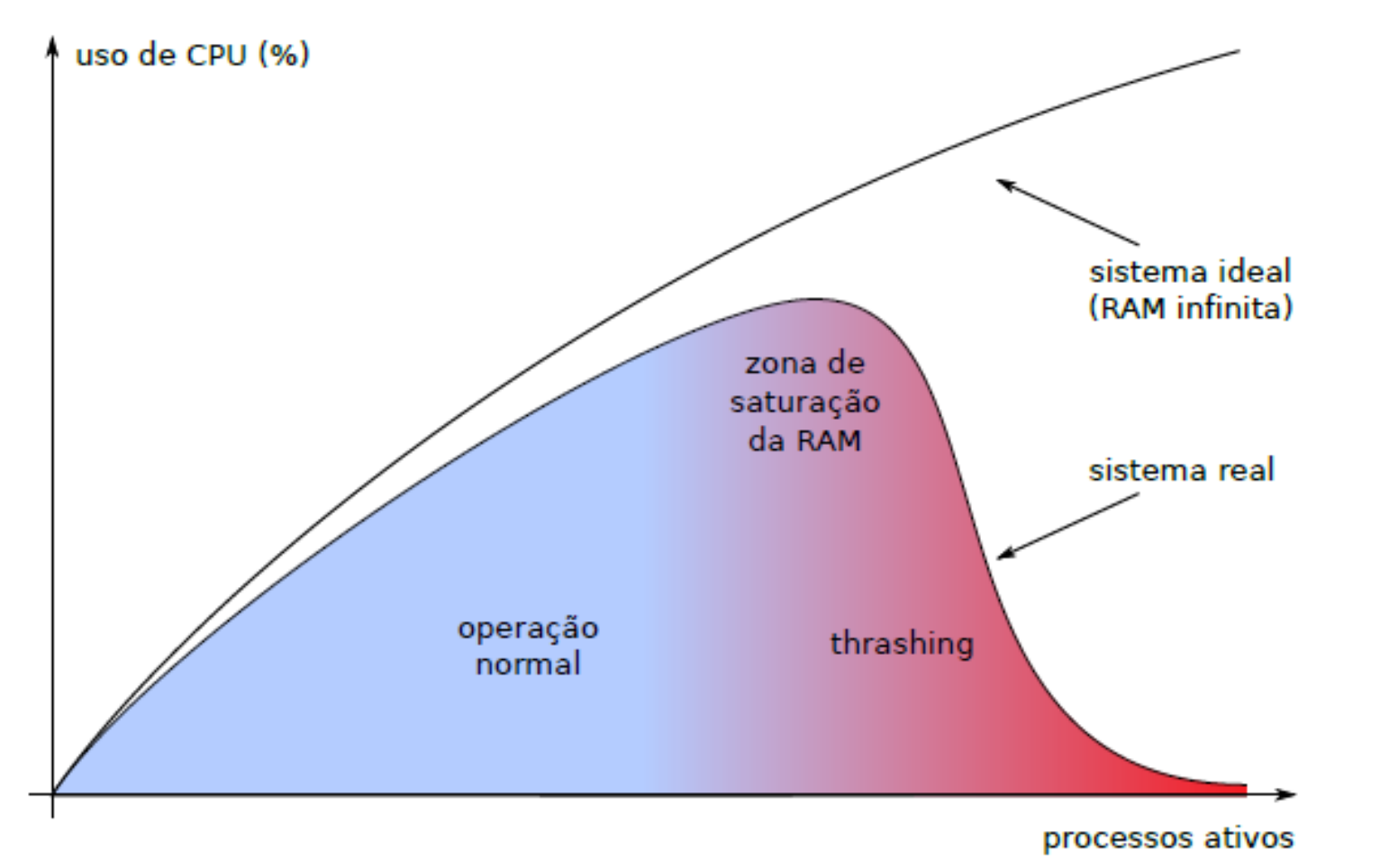
- **Qual o melhor pedaço para tirar?**
  - caso tirar um pedaço que logo seria executado, precisa ir buscá-lo novamente logo em seguida – **thrashing**
  - O SO tenta adivinhar, baseado no histórico, quais pedaços tem maior probabilidade de serem utilizados em seguida...

MEMÓRIA VIRTUAL - THRASHING

- Enquanto houver espaço na memória RAM para os processos ativos, não haverão problemas.
- No entanto, caso **o nº de processos prontos para execução seja maior que a memória RAM disponível no sistema**, poderá ocorrer *thrashing*.
  - memória RAM não é suficiente para todos os processos ativos

- Como todas as páginas estão em uso ativo, é necessário substituir uma página que vai ser necessária logo em seguida
- Quanto mais processos estiverem nessa situação, maior será a atividade de paginação e maior o número de processos no estado suspenso, aguardando páginas.

MEMÓRIA VIRTUAL - THRASHING



MEMÓRIA VIRTUAL - THRASHING

- Prejudica o desempenho do SO, a ponto de parar de responder ao usuário e se tornar inutilizável
- **Soluções:**
  - Aumentar a memória RAM
  - Limitar a quantidade máxima de processos ativos
  - Mudar a política de escalonamento dos processos durante o thrashing para evitar a competição pela memória disponível
  - Suspender em massa os processos ativos
  - Adotar uma política de escalonamento de CPU que considere o uso da memória
  - Abortar os processos com maior alocação de memória ou com maior atividade de paginação

MEMÓRIA VIRTUAL – PRINCÍPIO DA LOCALIDADE

- **Como o SO adivinha?**
  - baseado no **princípio da localidade -** um programa acessa uma porção relativamente pequena do espaço endereçável em um instante qualquer.

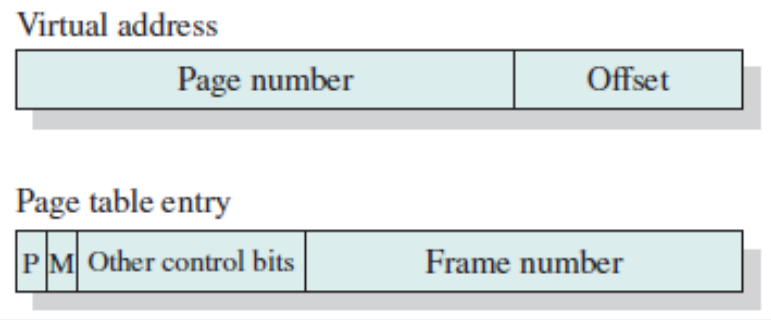
  - **Localidade Temporal** → Se um item é referenciado, ele tenderá a ser referenciado novamente. Exemplo: loops (instruções e dados).
  - **Localidade Espacial** → Se um item é referenciado, itens cujos endereços são próximos a este, tenderão a ser referenciados também. Exemplo: acesso a dados de um array.

MEMÓRIA VIRTUAL – IMPLEMENTAÇÃO

- Para a **memória virtual** ser **implementada** de forma **eficiente**, dois elementos são necessários:
  - deve existir **suporte de hardware** para uso de paginação ou segmentação
  - SO deve incluir software para **gerenciamento do movimento** das páginas ou segmentos entre memória principal e secundária.

MEMÓRIA VIRTUAL – PAGINAÇÃO

- Diferenças na implementação:
  - **Tabela de páginas se torna mais complexa**, já que apenas algumas páginas de um processo podem estar na memória principal
  - **Bit P em cada página** para indicar se está ou não presente na MP
  - **Bit M em cada página** para indicar se o conteúdo da página foi alterado desde que foi carregado para a MP
    - caso não houve mudança (ou seja, ela ainda não foi utilizada) não precisa ser substituída.



MEMÓRIA VIRTUAL – PAGINAÇÃO

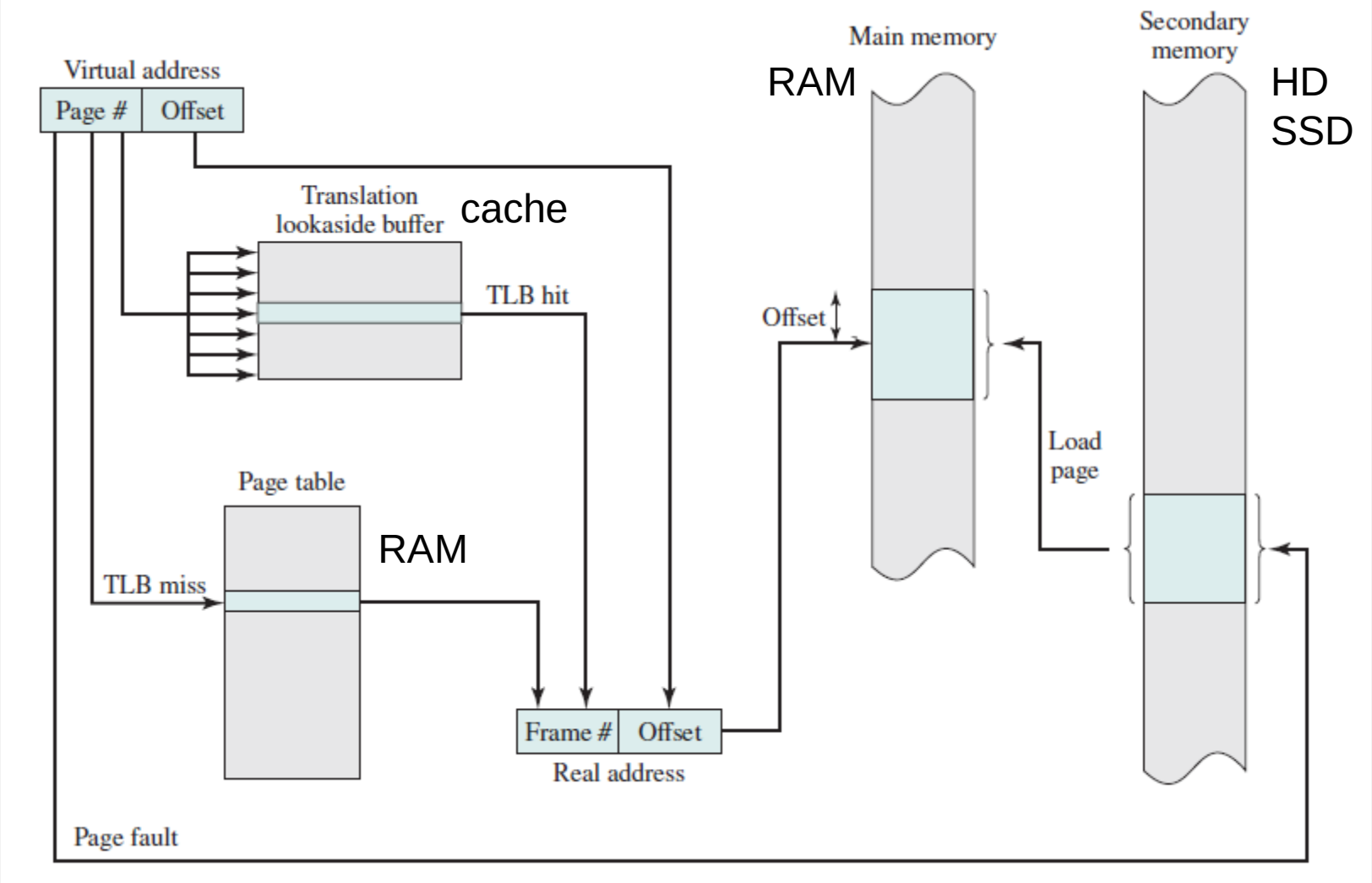
- Na paginação, cada acesso à memória virtual requer **dois acessos** a memória principal:
  - Um para buscar a tabela de páginas
  - Um para buscar os dados

- Acesso à memória é 2x mais lento, gerando perda significante de desempenho.
- **Solução:** utilizar hardware especial chamado de TLB

MEMÓRIA VIRTUAL – PAGINAÇÃO

- **TLB –** Translation look-aside buffer (TLB)
  - memória de alta velocidade (cache)
  - guarda as tabelas de páginas acessadas mais recentemente
  - Funcionamento:
    - Um endereçamento virtual é apresentado à MMU
    - Procura-se o endereço na TLB
    - O teste é feito em paralelo para todas as entradas
    - Se ele estiver presente, faz-se a tradução imediata
    - Senão (page fault) deve-se procurar na memória e atualizar a TLB

MEMÓRIA VIRTUAL – PAGINAÇÃO



MEMÓRIA VIRTUAL - EFICIÊNCIA

- **Memória virtual** permite usar o disco (memória secundária) como uma **extensão da memória RAM**, de forma transparente para os processos.

- Solução ideal caso não existisse uma limitação importante: **tempo de acesso a memória secundária.**
  - Cada falta de página → acesso a memória secundária
  - Faltas de páginas frequentes → muitos acessos a memória secundára
  - Aumento do tempo médio de acesso à memória → diminuição do desempenho geral do sistema.

MEMÓRIA VIRTUAL - EFICIÊNCIA

- A **frequência da falta de páginas** depende de vários **fatores**, como:
  - Tamanho da memória RAM em relação à demanda dos processos em execução – memória insuficiente ou muito carregada gera muitas faltas de páginas (*thrashing*)
  - Comportamento dos processos em relação ao uso da memória - agrupamento das páginas do processo, uso de menos páginas em simultâneo
  - Escolha das páginas a remover da memória - caso sejam removidas páginas usadas com muita frequência, estas serão provavelmente acessadas pouco tempo após sua remoção, gerando mais faltas de página.
    - Ver algoritmos de escolha das páginas a remover (a seguir).

GERENCIAMENTO DE MEMÓRIA

- A parte do SO responsável pela gerência de memória depende de três aspectos:
  - **Uso ou não de memória virtual**
  - **Uso de paginação, segmentação ou ambos**

  } depende do suporte de hardware

  - **Algoritmos utilizados para vários aspectos do gerenciamento**

POLÍTICA DE BUSCA DE PÁGINAS

- Determina quando uma página deve ser trazida para a memória principal

- **Demand paging**
- Uma página é trazida para a memória principal sob demanda (ou seja, quando acontece uma referência a uma localização naquela página)
- As páginas são trazidas para a memória seguindo o princípio da localidade

POLÍTICA DE SUBSTITUIÇÃO DE PÁGINAS

- Seleção de qual página será escolhida para ser substituída
- Objetivo: retirar a página com menos probabilidade de ser utilizada em um futuro próximo.
- Algoritmos:
  - **Ótimo**
  - **NRU – not used recently**
  - **LRU – least recently used**
  - **FIFO – first-in, first-out**
  - **Clock (relógio)**

POLÍTICA DE SUBSTITUIÇÃO DE PÁGINAS

- **Critérios de seleção -** das páginas a transferir da memória RAM para o armazenamento secundário
  - **Idade da página –** quanto tempo está na memória
  - **Frequência de acessos à página –** páginas muito acessadas tem chance de serem utilizadas novamente
  - **Data do ultimo acesso**
  - **Prioridade do processo proprietário –** alguns processos podem precisar de suas páginas rapidamente
  - **Conteúdo da página**

POLÍTICA DE SUBSTITUIÇÃO DE PÁGINAS

- A **escolha correta das páginas a retirar** da memória física é um **fator essencial para a eficiência** do mecanismo de paginação.

- Más escolhas poderão remover da memória páginas muito usadas, aumentando a taxa de faltas de página e diminuindo o desempenho do sistema.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo Ótimo**
  - seleciona para substituir a página que vai ficar mais tempo sem ser usada
  - Considera as páginas que estão em memória:
    - Para cada uma, determinar quantas instruções faltam para que a página seja acessada novamente
    - Significa determinar o fluxo dos programas e quantas instruções ainda serão executadas antes que cada página seja acessada de novo
    - Ou seja, determinar se cada página será acessada em 10, 100 ou 1000 instruções
  - Melhor algoritmo, mas impossível de implementar (teria que prever o futuro)

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo NRU – not used recently**
  - 2 bits associados com cada página
    - R – Referencia (R ou W)
    - M – Modificada (W)
  - Esses bits são "setados" toda vez que uma página é acessada
    - Mantidos por hardware
    - A cada interrupção de tempo, todos os bits R são zerados
      - Distinguir as páginas referenciadas recentemente pelo processo em execução

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo NRU – not used recently**
  - Em um page fault, todas as páginas são divididas em 4 categorias
    - Classe 0: não referenciada, não modificada (R = 0, M = 0)
    - Classe 1: não referenciada, modificada (R = 0, M = 1)
    - Classe 2: referenciada, não modificada (R = 1, M = 0)
    - Classe 3: referenciada, modificada (R = 1, M = 1)

  - **Obs:** Classe 1 ocorre quando uma página classe 3 tem o bit R zerado numa interrupção de tempo

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo NRU – not used recently**
  - O algoritmo remove aleatoriamente uma página da classe mais baixa
  - **Importante**: É preferível remover uma página modificada do que uma referenciada com frequência
  - **Vantagens:**
    - Fácil de implementar e entender
    - Apresenta um bom desempenho

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo LRU – least recently used**
  - Funcionamento:
    - A página que não foi referenciada (usada) há mais tempo é substituída
  - Ideia:
    - Páginas muito usadas nas últimas instruções serão muito usadas nas próximas
    - Páginas não usadas há muito tempo provavelmente permanecerão assim

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

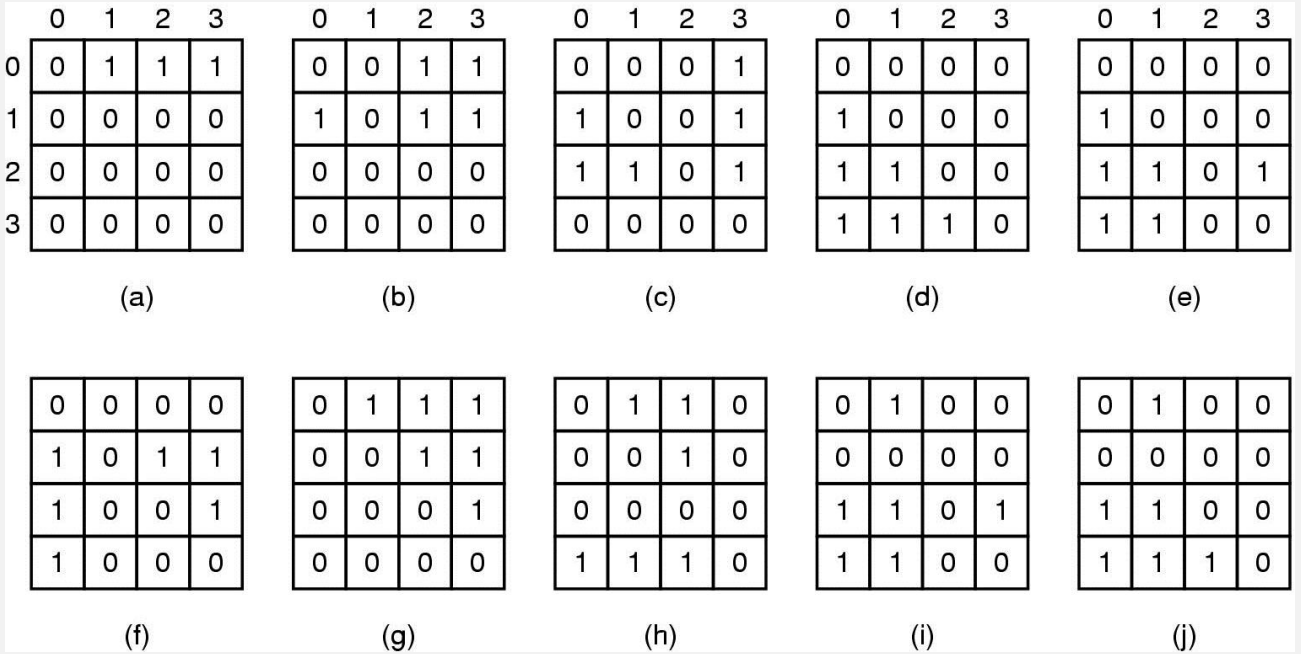
- **Algoritmo LRU – least recently used**
  - Problema: dificuldade de implementação
    - Manter uma lista com informação para cada página de quando foi sua última referência
    - Isto precisaria ser feito a cada acesso de memória (processo muito custoso a nível de processamento)

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo LRU – least recently used**
  - Solução:
    - Para uma máquina de n frames, uma implementação em hardware pode manter uma matriz de n x n bits
    - Em acesso a uma página:
      - Todos os bits da linha correspondente a página são setados (colocado o valor 1);
      - Todos os bits da coluna são zerados
    - A linha com menor valor binário corresponde à página LRU

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo LRU – least recently used**
  - **Exemplo**: um sistema com 4 páginas (0-3)
  - As seguintes páginas são acessadas: 0 1 2 3 2 1 0 3 2 3

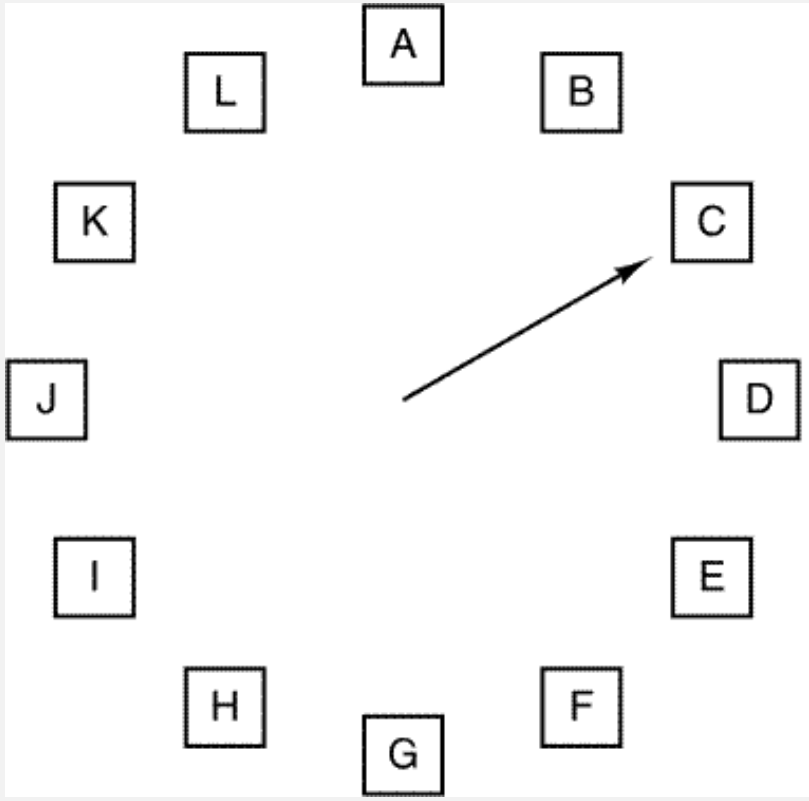


ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo FIFO – first-in, first-out**
  - Funcionamento
    - O SO mantém uma lista ordenada de páginas, sendo que a primeira é a mais antiga na memória
    - Quando chega uma página nova, ela é inserida no fim da fila
    - Se é preciso descartar uma página, a mais antiga é escolhida
  - Problemas:
    - Não considera a importância das páginas
    - Descarta páginas antigas, mas que podem ser necessárias
    - Mantém as páginas mais novas, mas que talvez não tenham importância

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- **Algoritmo Clock (relógio)**



Quando ocorre uma falta de página,
a página apontada é examinada.
A atitude a ser tomada depende do bit R:
R = 0: Retira a página,
R = 1: Faz R = 0 e avança o ponteiro.

EXERCÍCIO

- Considere um sistema de memória com quatro frames de RAM e oito páginas a alocar.
- Os frames contêm inicialmente as páginas 7, 4 e 1, carregadas em memória nessa sequência.
- Determine quantas faltas de página ocorrem na sequência de acesso {0, 1, 7, 2, 3, 2, 7, 1, 0, 3}, para os algoritmos de escalonamento de memória FIFO, ÓTIMO e LRU.

EXERCÍCIO - RESPOSTA

- Considere um sistema de memória com quatro frames de RAM e oito páginas a alocar.
- Os frames contêm inicialmente as páginas 7, 4 e 1, carregadas em memória nessa sequência.
- Determine quantas faltas de página (FP) ocorrem na sequência de acesso {0, 1, 7, 2, 3, 2, 7, 1, 0, 3}, para os algoritmos de escalonamento de memória FIFO, ÓTIMO e LRU.

FIFO – 6 FP **Ótimo – 4 FP** **LRU– 5 FP**

PRÓXIMA AULA

- Revisão Gerenciamento de memória e Exercícios

BIBLIOGRAFIA

- Tanenbaum, A. S. **Sistemas Operacionais Modernos.** Pearson Prentice Hall. 3rd Ed., 2009.
- Silberschatz, A; Galvin, P. B.; Gagne G.; **Fundamentos de Sistemas Operacionais.** LTC. 9th Ed., 2015.
- Stallings, W.; **Operating Systems: Internals and Design Principles**. Prentice Hall. 5th Ed., 2005.
- Oliveira, Rômulo, S. et al. **Sistemas Operacionais** - V11 - UFRGS. Disponível em: Minha Biblioteca, Grupo A, 2010.