Universidade Federal de Uberlândia
Faculdade de Computação
Sistemas Operacionais

Memória Virtual

Prof. Dr. Marcelo Zanchetta do Nascimento

Roteiro

- Introdução: O que é Memória Virtual?
- Paginação por demanda
- Cópia após gravação
- Substituição de páginas
- Thrashing
- Conjunto de trabalho
- Considerações
- Exemplos
- Exercícios
- Leituras Sugeridas

Introdução

Memória Virtual (MV)

- **Ideia básica:** tamanho total de um programa pode exceder a quantidade de memória física disponível em um SC;
- Essa técnica que combina memória principal (RAM) e memória secundária (disco rígido HDD ou SSD);
- Fundamentação:
  - não vincular o endereçamento feito pelo programada ao endereço físico da memória principal.
- Busca maximizar o número de processos na memória RAM.

Introdução

Memória Virtual (MV)

- Vantagens:
  - Busca reduzir a fragmentação na memória;
  - Permite trabalhar com **estruturas de dados maiores** que o tamanho da memória RAM;
- O espaço de endereçamento da memória RAM e a memória secundária tem a estrutura de blocos do mesmo tamanho:
  - As páginas no espaço virtual: denominados páginas virtuais;
  - As páginas em memória RAM: páginas reais ou frames (quadros).

Introdução

- Permite que um programa não precise estar em endereços contíguos na memória RAM para ser executado;
- Em sistema sem memória virtual, o endereço virtual é colocado diretamente no barramento de memória.

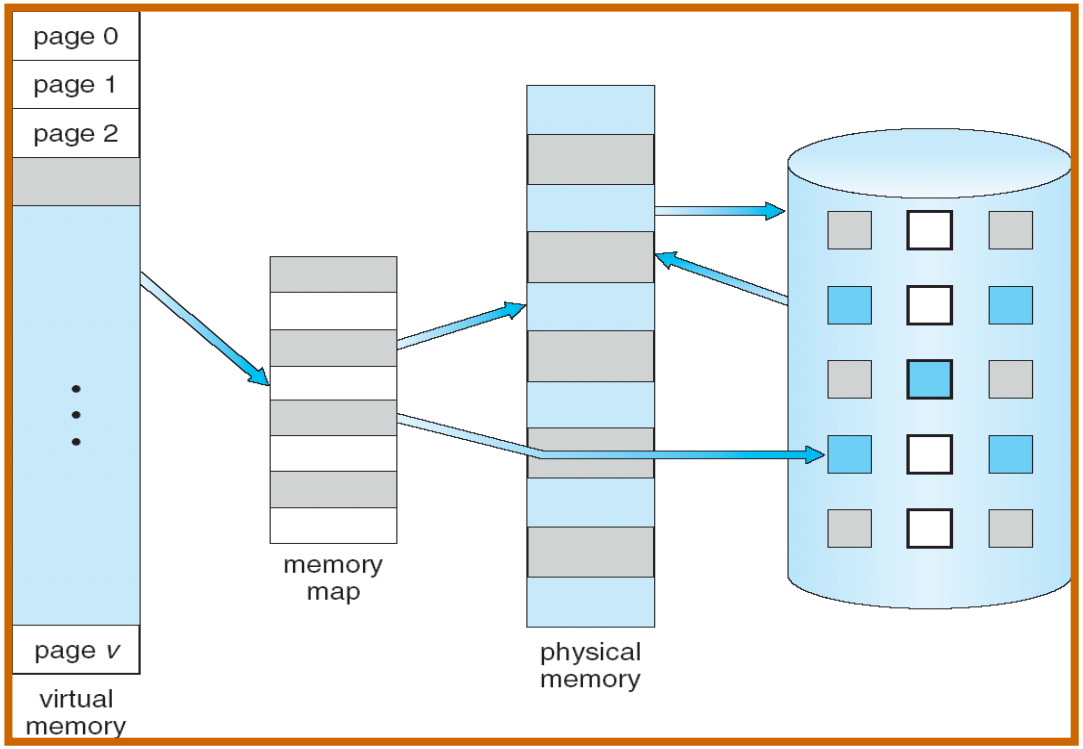


Figura 1. Representação de endereço virtual e endereço fisico

Introdução

- No ambiente Linux, para verificar dados da memória de um processo, deve-se empregar:
  - *cat /proc/meminfo*
  - *cat /proc/<número processo>/maps*
  - *pmap -x <número processo> |more*
  - *free -m*
  - *top*
  - */proc/sys/vm (Sistema de arquivo no Linux)*
  - *getconf PAGESIZE*

Introdução

Tabela de páginas: responsável por mapear páginas virtuais em quadros de páginas para cada processo;

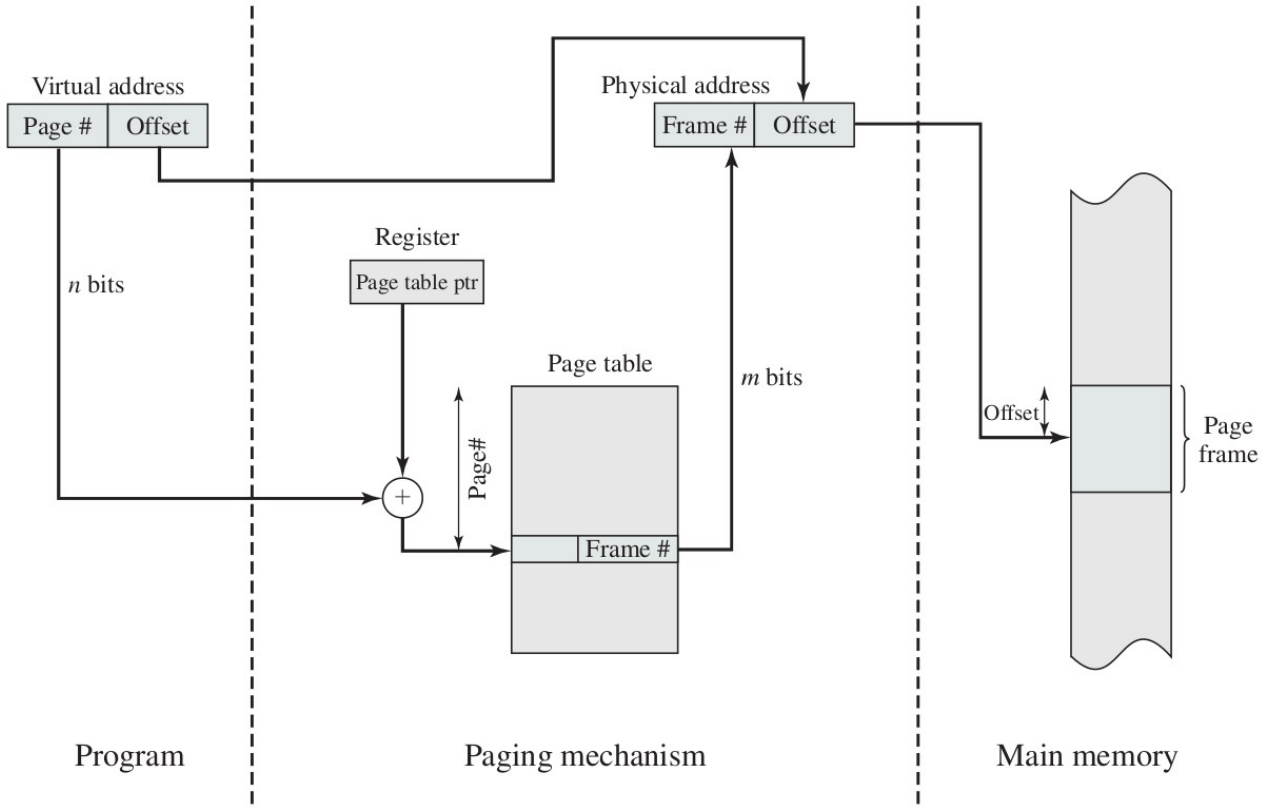


Figura 2: Tradução de endereço em um sistema de páginas na memória

Introdução

As políticas de busca de página são baseadas nas seguintes estratégias:

Paginação por demanda: Páginas transferidas da memória secundária para a principal quando referenciadas:

- É possível que partes não executadas do programa nunca sejam carregadas.

Paginação antecipada: Além da página referenciada, carrega páginas que podem ou não ser necessárias.

- Essa estratégia permite economia de tempo, mas pode causar desperdiço com dados não empregados pelo processo.

Paginação por demanda

- Uma página é carregada para memória apenas quando necessária.
- Motivo:
  - Necessário menor número de chamadas de E/S;
  - Resposta mais rápida, porque carrega apenas as primeiras páginas do processo;
  - Menos espaço de memória para cada processo;
  - Um número maior de processos admitidos na memória.
- Como isso acontece?
  - Processo gera o endereço lógico (virtual), o qual é mapeado para o endereço físico;
  - Se a página solicitada não está na memória RAM, o SO carrega do disco (memória secundária).

Paginação por demanda

Onde estão os dados de um processo?

- Cada entrada na tabela de página de um processo tem um bit "**valid–invalid**":
  - Inicial: parte configurado com i;
  - **V:** presente na memória física;
  - **I:** não está na memória física;
- Durante a tradução de endereço, se o bit "valid–invalid" esta com **i,** pode ser:
  - **Referência ilegal** (espaço de endereçamento fora do espaço de endereçamento do processo): aborta processo;
  - **Referência legal:** mas não está na memória RAM → gera o problema ***page fault*** (deve carregar a página do disco para a RAM).

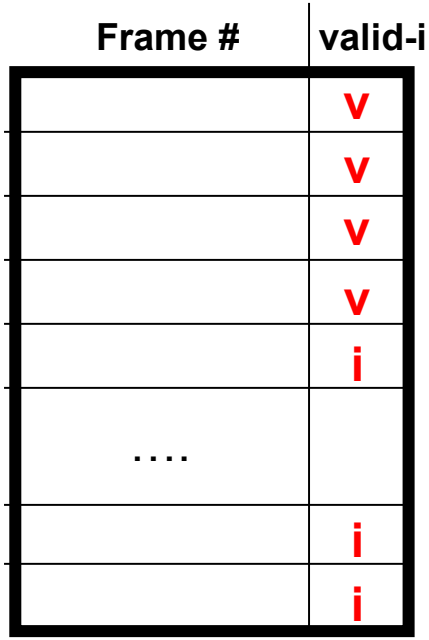


Tabela de Páginas de um processo

Paginação por demanda

- **O que ocorre se uma página não esta na memória?**
  - O acesso a uma página marcada como inválida (i) causa uma trap (interrupção) do tipo **"page fault".**
- Essa trap é repassada ao SO e ocorre um **procedimento** para tratá-la com objetivo de decidir:
  - referência inválida: aborta o processo;
  - não está na memória: carrega a página:

    1) encontrar um quadro livre na RAM;

    2) trocar a página do disco para o quadro (Operação de E/S);

    3) modificar a tabela de páginas com o bit válido (**v**)**;**

    4) reiniciar a instrução que causou a trap page fault.

Paginação por demanda

Como esse procedimento é realizado no sistema?

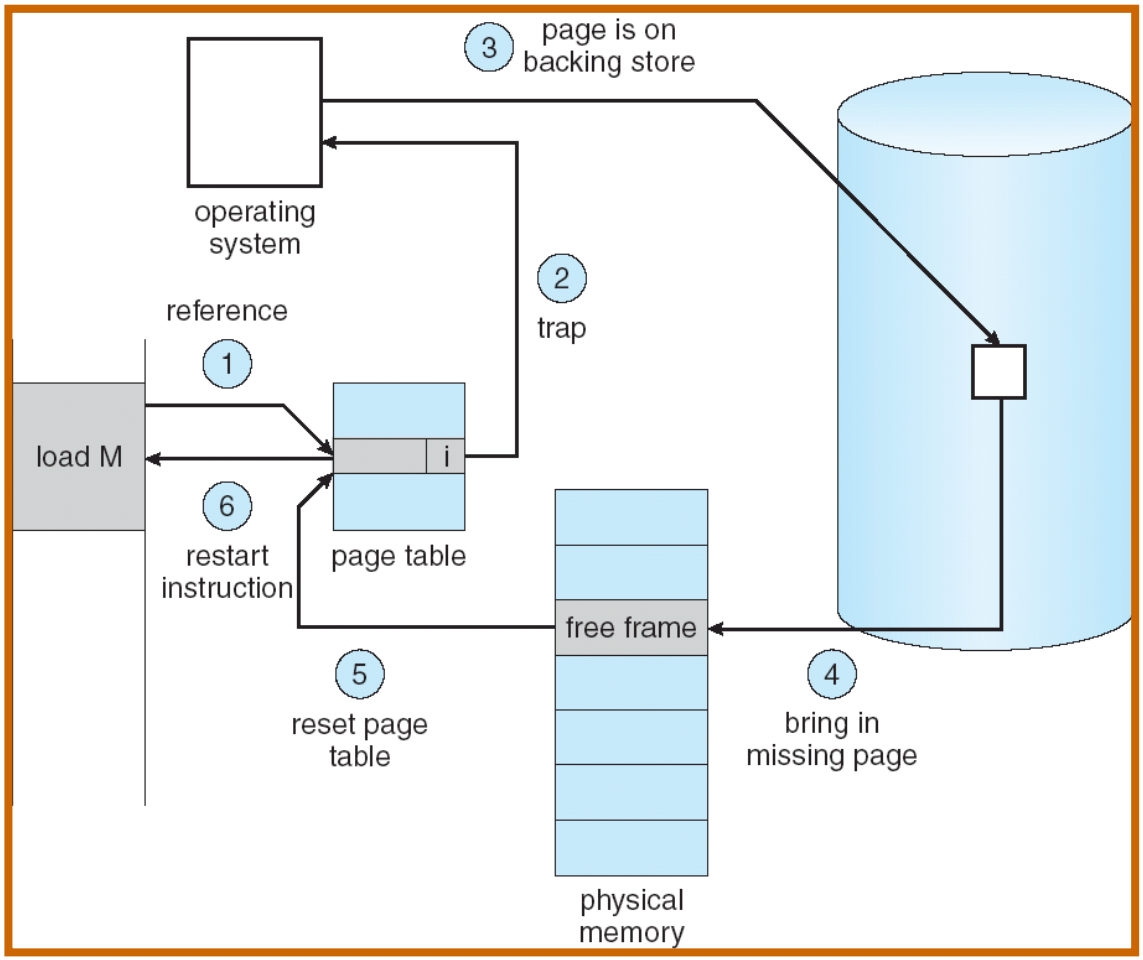


Figura 3. Atende requisição de uma página que não está na memória RAM.

Paginação por demanda

Desempenho da paginação por demanda:

- Deve medir a probabilidade de uma falha de página $0 \leq p \leq 1.0$
  - Se $p = 0$ significa que não houve falha de página;
  - Se $p = 1$ significa que toda referência é uma falha.

- Avaliar o **Tempo de Acesso Efetivo** (TAE):

  TAE = $(1 - p)$ * Tempo de acesso a memória (10 a 200 nanossegundos) + p * tempo de falha de página.

  - Tempo de falha de página: representa o serviço de interrupção de falha de página (~microsegundos) + Leitura da página requerida (~milissegundos) + reinício do processo (~microsegundos).

- Lembre-se: Leitura de uma página requerida pode solicitar escrita de outra página para o disco se não houver quadro livre.
- Então, esse tempo pode ainda ser maior para essa tarefa.

Cópia após gravação

Essa técnica permite que ambos os processos, pai e filho, compartilhem, inicialmente, as mesmas páginas na memória (Exemplo: system call fork( ) - empregada no Unix):

- Minimiza o número de novas páginas a ser alocadas aos processos recém-criados;
- Chamada fork( ): cria uma cópia do espaço de endereços do pai para o processo filho;
- Se algum processo modificar uma página compartilhada - a página é copiada;
- Apenas as páginas que puderem ser modificadas serão copiadas;
- Pai e filho compartilham as páginas não modificadas.

Cópia após gravação

- O SO usa uma cadeia de páginas livres (**pool**), a qual é alocada quando um processo precisa ser expandido;
- Fornece uma cadeia de páginas livres para a solicitação;
- Usa a técnica **preenher-com-zero-sob-demanda** em que páginas são zeradas antes de serem alocadas.

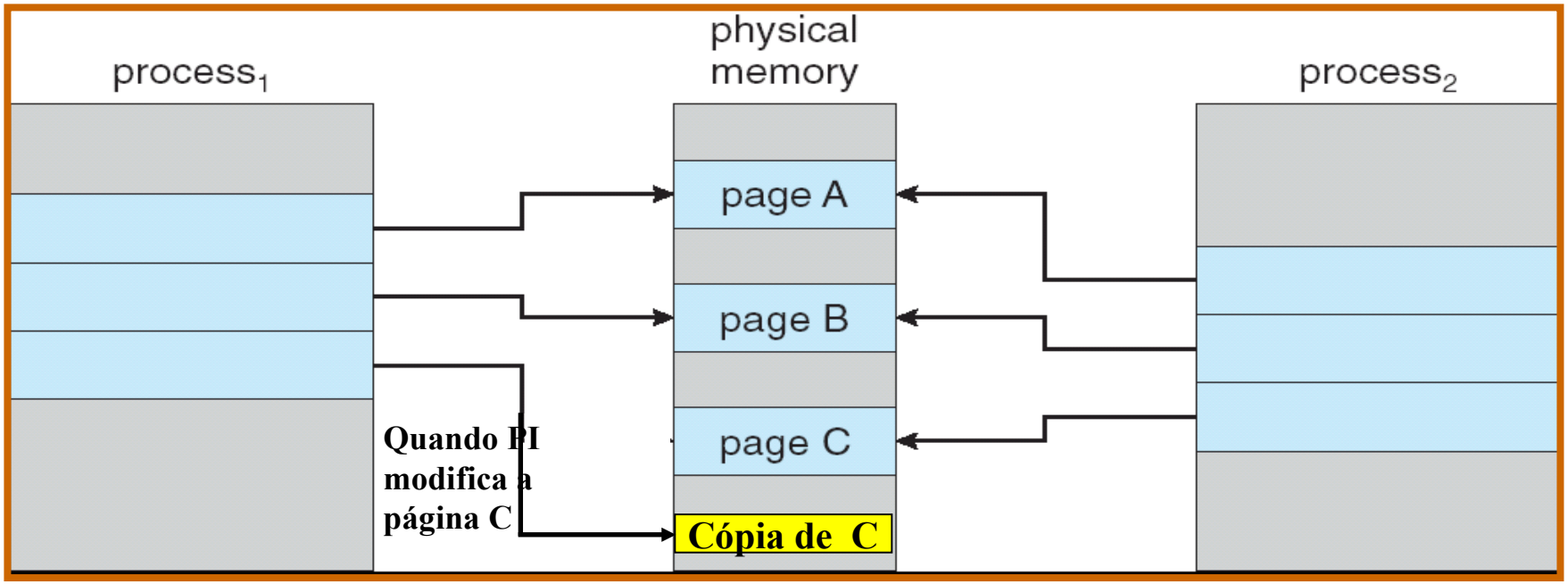


Figura 4. Após o processo 1 modificar a página C tem-se uma cópia.

Substituição de Páginas

- A falta de página em memória RAM e a necessidade de carregar uma página do processo requerida do disco.
- Exige espaço disponível em memória RAM.
- Então:
  - Encontrar a localização da página no disco;
  - Determinar uma página livre:
    - Se há um quadro livre é só usá-lo;
    - Senão deve selecionar uma página “vítima” residente na memória.
  - Carreguar a página requerida para o quadro livre;
  - Atualizar a tabela de página e a lista de quadro livre;
  - Reiniciar o processo.

Substituição de Páginas

- Mas e o problema de sobrecarga? Escolher a página vítima que não foi modificada (reduz operação de E/S).
- Usa um **bit de modificação** com cada página para indicar se a página foi modificada.
- E quando todas foram modificadas, como escolher a página vítima?

Algoritmo de substituição de página

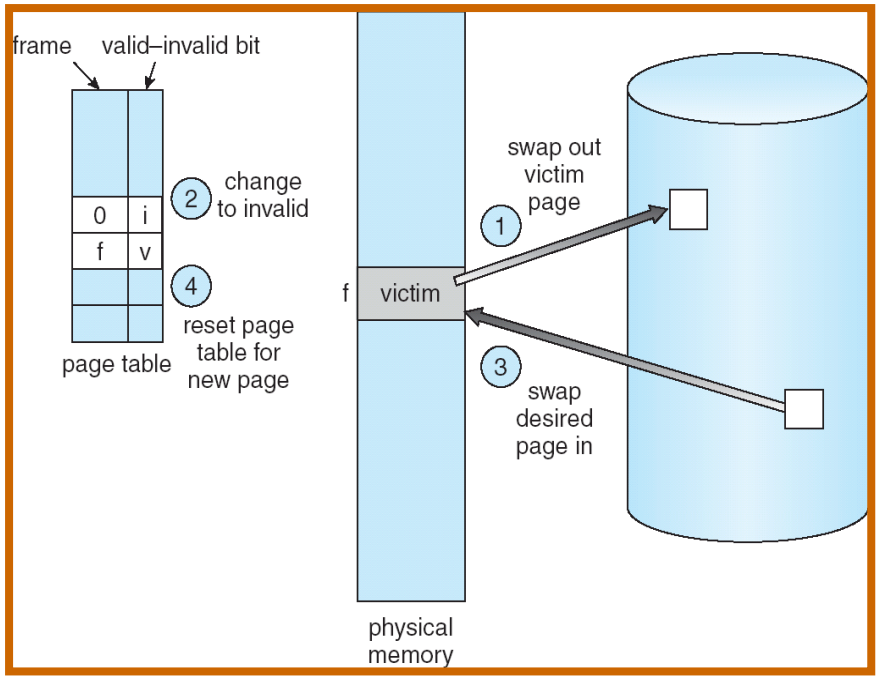


Figura 5. Necessidade de uma página ser substituida

Substituição de Páginas

- Objetivo: **minimizar** a taxa de falta de página (page fault);
- Deve ter um algoritmo:
  - Retira um conjunto de referência da memória, denominado **string de referência,** e calcula o número de falha de página na string.
- Dado uma string de referência: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**
- Em que usa-se o número de página:
  - Exemplo:
    - O endereço das páginas em sequência: 100, 250, 270, 300 (Assumindo uma página de tamanho de 100 bytes).
    - As referências 250 e 270 estão na mesma página (página número 2), **apenas a primeira referência** pode causar uma falha de página (a diferença esta interna no offset – deslocamento na página).

Substituição de Páginas

Algoritmo: First-in First-out

String de referência: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

- 3 Quadros (3 páginas na memória RAM);

Trabalha:

- Em toda falha de página: verifica o conteúdo da memória;

Vantagem:

- Fácil de entender e implementar.

Desvantagem:

- Desempenho pode não ser adequado: substitui uma página usada constantemente:
- Exemplo:
  - Uma variável acessada constantemente.

Substituição de Páginas

Algoritmo: First-in First-out

Exemplo: Tem-se a seguinte string de referência:

- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	4	5	**9 falhas ocorrem neste**
2	2	1	3	**algoritmo**
3	3	2	4	

Substituição de Páginas

First-in First-out

- Com **3 quadros:** Quantas falhas de páginas pode ocorrer?
  - **9** falhas de páginas (3 primeiros causam erro de página)
- Com **4 quadros:** Quantas falhas de páginas?
  - **10** falhas de páginas
- Mais quadros podem ser considerados problema em falha de página.
  - Pode ocorrer a **anomalia de Belady**.
  - Anomalia Belady:
    - mais quadros - mais falha de página.

Substituição de Páginas

First-in First-out

- **Anomalia de Belady** – falta de 10 páginas para 4 quadros e 9 páginas para 3 quadros.
- Em alguns algoritmos, a taxa de erro de página aumenta com o número maior de quadros do sistema computacional.

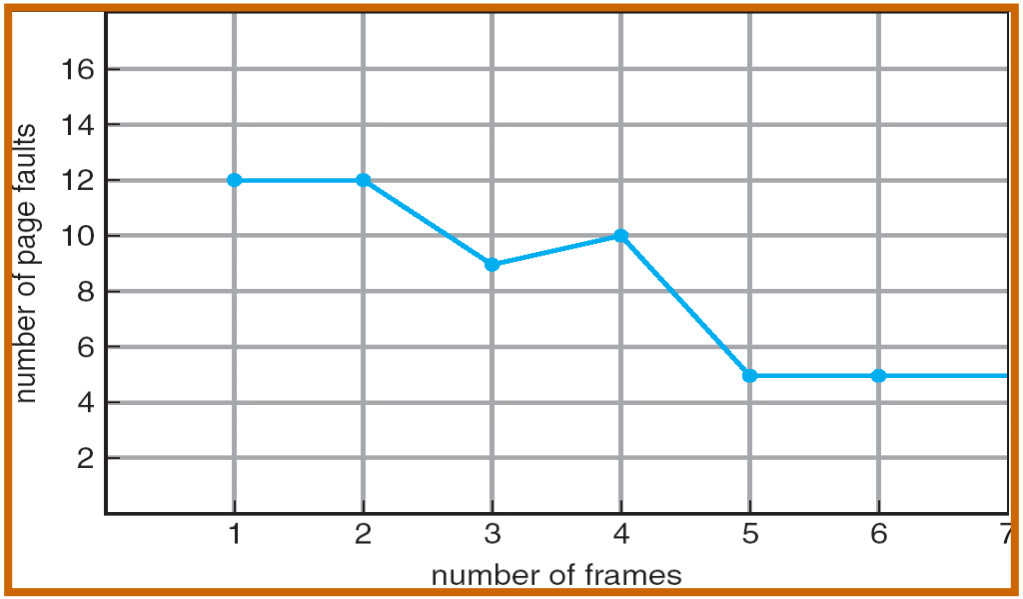
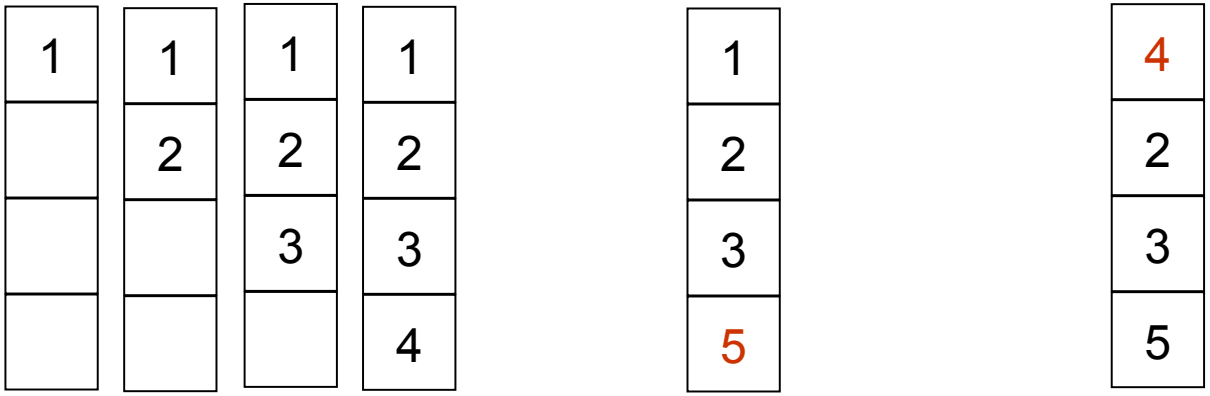


Figura 6. Número de páginas versus falha de páginas

Substituição de Páginas

Algoritmo ótimo:

- Você pode querer um algoritmo de substituição ótimo?
  - Troca de página não irá usar o mais longo periodo de tempo, como por exemplo, o FIFO.
- Exemplo: 4-quadros: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

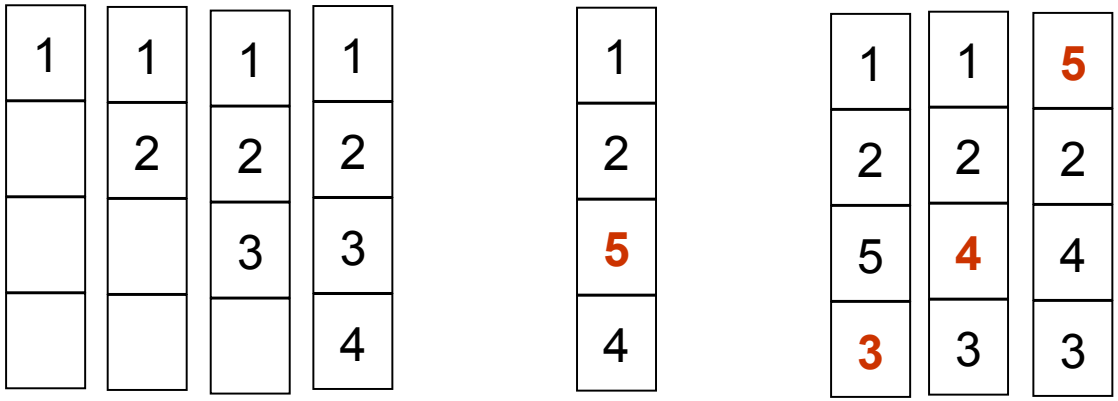


- 6 falhas de página
- **Como prever o futuro? Não é possível**

Substituição de Páginas

Algoritmo “Least Recently Used” (LRU)

- Tentativa de uma aproximação a política ótima: “olhe no passado para decidir o futuro”.
  - LRU: Troca a página que não tem sido utilizada por o mais longo período;
  - Relação: essa página pode não ser necessária (exemplo: páginas de inicialização de um módulo);
- Exemplo: memória com 4 quadros e requisições 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4 e 5



- Quantas falhas ocorrem se usar o FIFO?

Substituição de Páginas

Como implementar o LRU? Implementação 1: Contadores

- Toda entrada na tabela de páginas tem um **campo de tempo** de uso (um contador);
- Quando a página é referenciada, cópia o tempo do relógio da CPU para esse campo;
  - O tempo da CPU é mantido em um registrador e incrementado com todo acesso a memória;
- Necessário trocar uma página, busca a página com o tempo mais antigo;
- **Problema**:
  - Buscar o "tempo" em CPU, atualizar o campo a cada uso de memória (escreve na memória).
- Necessário suporte de hardware do Sistema Computacional.

Substituição de Páginas

LRU - Implementação 2: Pilha

- Mantém uma **pilha** do número de páginas em um lista duplamente encadeada;
- Se uma página é referênciada, move para o topo;
- A página menos recentemente usada vai para parte inferior;
- Via hardware ocorre a atualização da pilha.
- Deve manter na memória a pilha de páginas referenciadas,

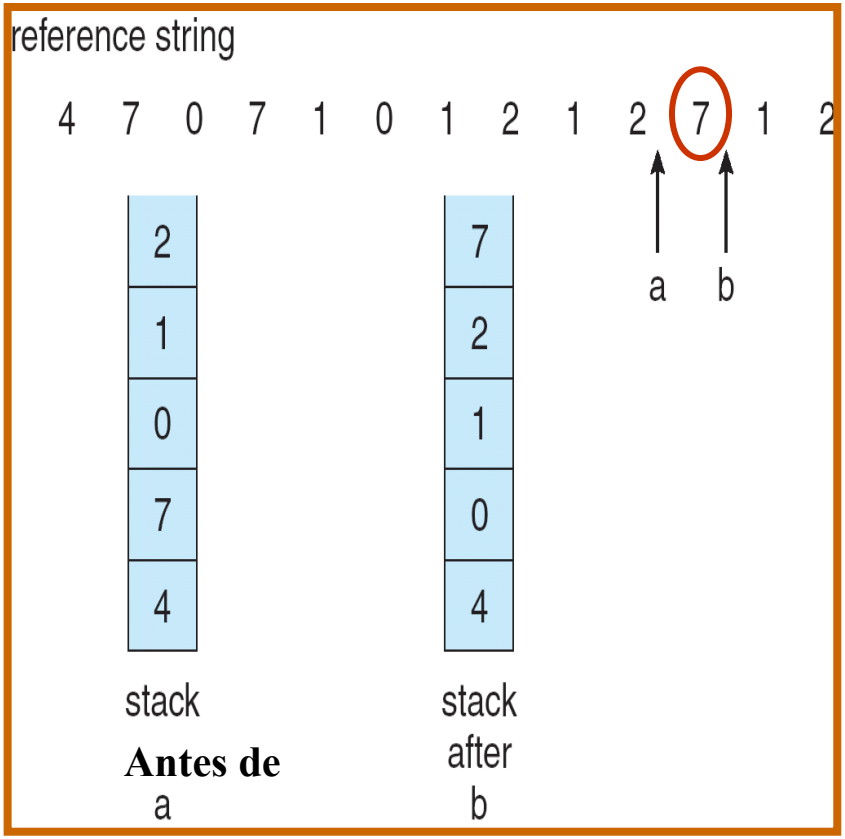


Figura 7. Uso de uma pilha para o registro da 7 referência da página mais recente.

Substituição de Páginas

Aproximação de LRU: bits de referência

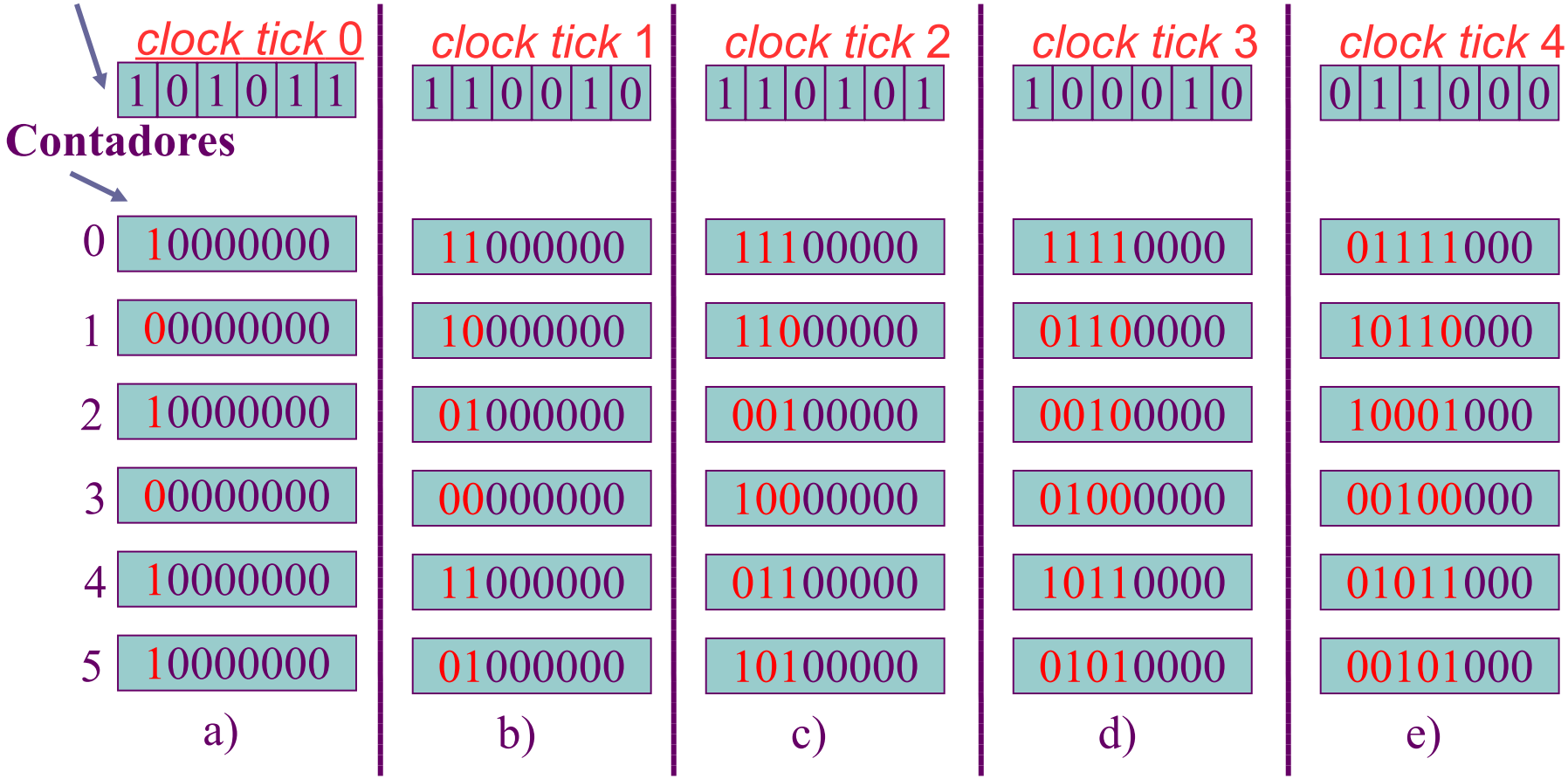
- Pode ser implementado por Hardware
- Um campo “extra” para armazenar o valor do bit de referência na tabela de páginas. Após cada referência à memória, o valor igual a 1 é armazenado nesse campo adicional;
- Quando ocorre falta de página, o SO examina os campos da tabela de página a fim de encontrar o que não foi usado;
- Qual a ordem de uso (como foi referênciado) do bit?
- Essa solução não tem uma ordem de uso das páginas.
- Então, uma solução é ter mais bits associados a essa entrada.

Substituição de Páginas

Aproximação LRU: Uso de um Byte (8 bits)

- Solução: Armazena o histórico de uso para os últimos 8 clocks (1 Byte). Conhecido como **algoritmo do Aging (envelhecimento).**

Exemplo: bits de referência para as páginas 0 - 5



Substituição de Páginas

Algoritmo Segunda Chance (bits)

- Uma outra aproximação do algoritmo LRU;
- Cada página tem um bit de referência, inicialmente, com valor igual a 0;
- Quando a página é referenciada “ref_bit = 1” (pelo hardware)
  - Mantém um ponteiro para próxima vítima (candidato);
- Quando escolhe uma página para substituir, verifica o ref_bit para escolha da “vítima”:
  - Se ref_bit == 0, então, troca a página
  - Senão configura o ref_bit = 0
    - Deixa a página na memória (segunda chance),
    - Move o ponteiro para próxima página,
    - Repete até encontrar uma vítima.

Substituição de Páginas

Segunda Chance

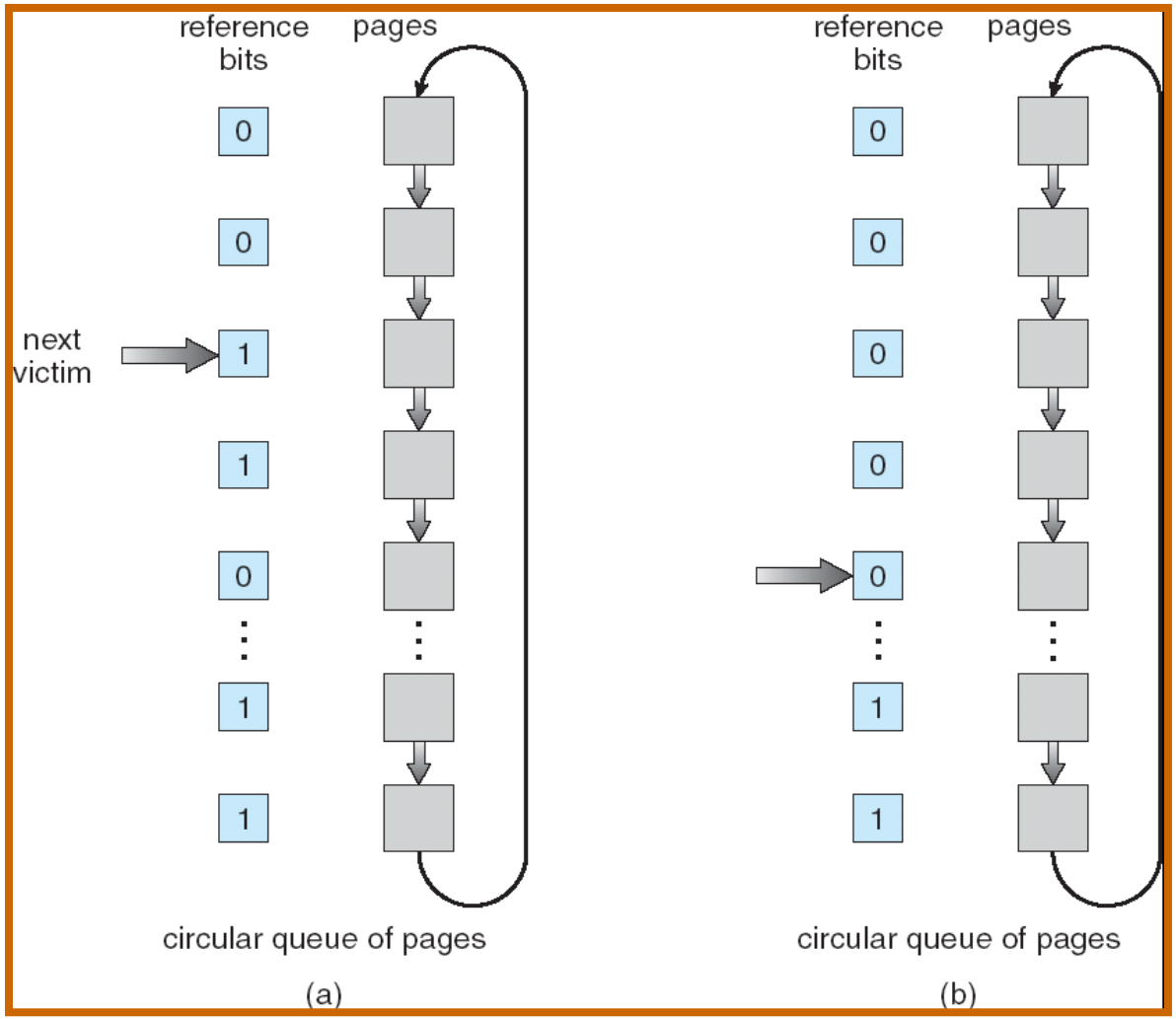


Figura 8. Algoritmo de substituição da segunda chance

Substituição de Páginas

Algoritmo Segunda Chance “Melhorado”:

- Considere os bits R (referência) e M (modificado) como pares ordenados;
- O SO inspeciona todas as páginas e separa em 4 categorias, com base nos valores desses bits:
  - Classe 0 => não referenciada, não modificada (0, 0);
  - Classe 1 => não referenciada, modificada (0, 1);
  - Classe 2 => referenciada, não modificada (1, 0);
  - Classe 3 => referenciada, modificada (1, 1);
- Usa o mesmo esquema do relógio (segunda chance);
- Examina a classe à qual pertende e substitui a primeira página encontrada na classe mais baixa.

Substituição de Páginas

Algoritmos baseados em contadores (1):

- Guarda um contador com o número de referência que tem ocorrido em cada página;
- O algoritmo LFU (Least Frequently Used): retira página com menor contador:
  - Página com o menor valor para o contador não é usada frequentemente;
  - Problema: se alguma página for usada muito frequentemente no início do processo, mas não for mais usada irá permanecer na memória.
  - Exemplo:
    - Programa com matriz de dados irá manter a página com esses dados na memória.

Substituição de Páginas

Algoritmos baseados em contadores (2):

- Algoritmo MFU (Most Frequently Used): retira a página com mais alto valor de contador.
  - Usa o argumento de que a página com menor contagem é possível que acabou de chegar na memória e ainda não está em uso.
  - Problema dessa abordagem:
    - O código de uma subrotina usado raramente;
    - Esse algoritmo considera um bom candidato e não será eliminado.
    - Então, pode manter uma subrotina muito pouco usada no código.

Thrashing

- O que ocorre quando um processo não tem quadros suficientes para manter o conjunto de páginas ativas na memória?

- Ocorre uma taxa de falha de página muito alta:
  - Então, há baixo uso de CPU;
  - Isso faz com que o SO “pense” que é preciso aumentar o grau de multiprogramação e enviar mais informações;
  - O SO admite outros processo ao sistema (piorando a situação);
- Então, ocorre o **thrashing:** uma excessiva transferência de páginas entre a memória principal e a memória secundária.
  - Problema existente tanto em paginação quanto em segmentação.

Thrashing

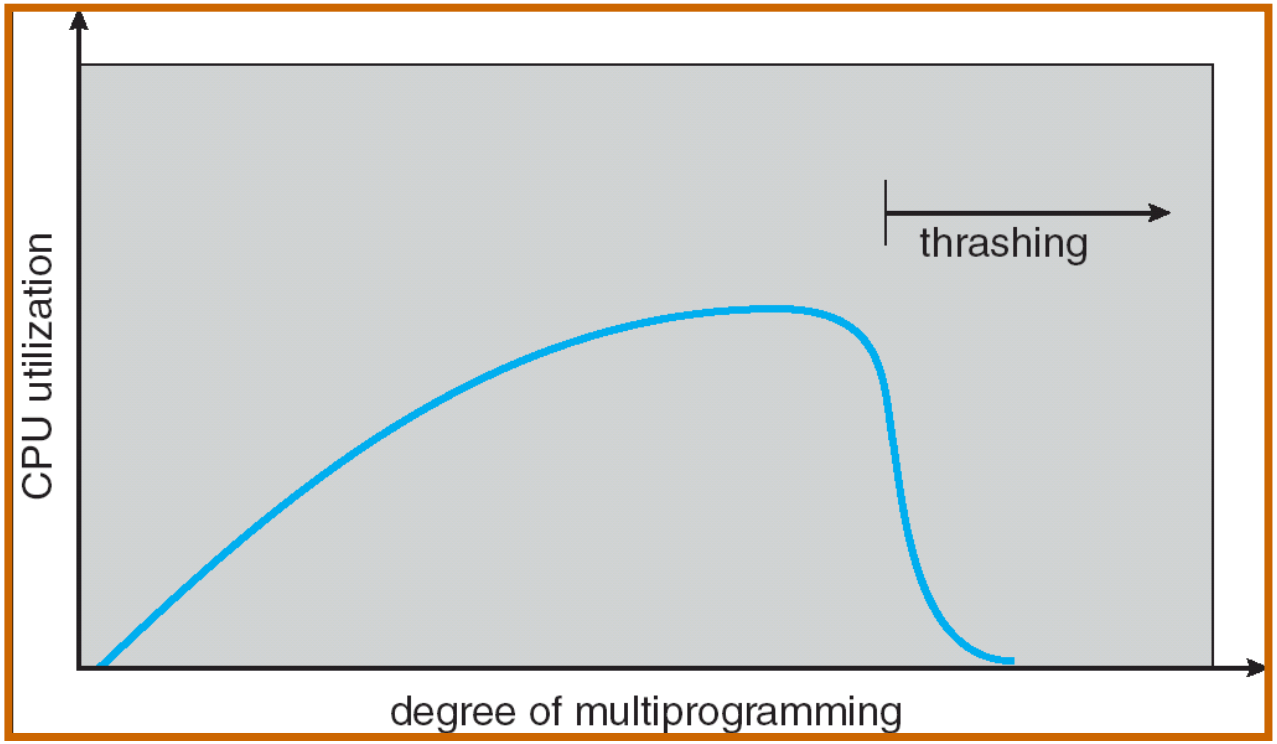


Figura 9. Atividade improdutiva (thrashing)

Thrashing: Como tratá-lo em S.O.?

- Para previnir esse problema, deve-se fornecer a cada processo o número de quadros minímos.
- Mas como saber quantos quadros um processo precisa?
  - Um programa é composto por várias funções ou módulos;
  - Quando executa uma função, as referências a memória são feitas para as instruções e variáveis locais da função e variáveis globais;
  - Então, pode ser necessário guardar em memória as páginas necessárias para executar a função;
  - Após finalizar a função, pode-se executar outra e trazer as páginas necessárias para a nova função.
  - Isso é conhecido como **"modelo de localidade";**

Modelo de Localidade

- O "**estado**" do modelo de localidade:
  - Quando um processo é executado move de uma localidade para outra (um conjunto de páginas que são ativamente usada naquele momento).
- Lembre-se:
  - Localidade não é restrito apenas aos módulos e funções.
  - Pode ser um segmento de código da função: instruções em várias páginas;
  - Localidade de um programa pode sobrepor-se;
  - Localidade contribui para o sucesso de paginação por demanda;
- Como podemos saber o tamanho de uma localidade?
- Usando o modelo de conjunto de trabalho (Working Set);

Conjunto de Trabalho

- O conjunto de páginas mais referenciadas na memória Δ:
  - Cria uma janela do Working Set (WS);
  - Em cada referência, uma página é adicionada ao conjunto, senão for mais usada, sai do conjunto;
  - Exemplo: Δ = 10 (tamanho da janela);
    - Tamanho do WS em tempo *t1* é igual a 5 páginas;
    - Em tempo *t2* é igual a 2 páginas.

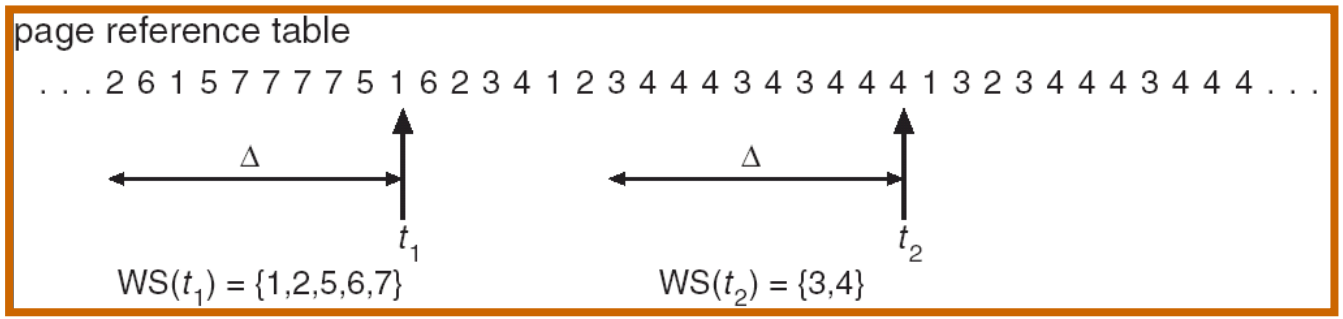


Figura 10. Modelos de WS para páginas referenciadas.

Conjunto de Trabalho

- A precisão do modelo WS depende da escolha de Δ
  - Se Δ é muito pequeno, não abrangerá a localidade inteira;
  - Se Δ é muito grande, poderá sobrepor várias localidades;
  - Se $\Delta = \infty \Rightarrow$ um conjunto com todas as páginas referenciadas durante a execução.

- Usando modelo WS:
  - O SO monitora o WS de cada processo;
  - Ele aloca um número de quadros suficiente para fornecer o tamanho do seu conjunto de trabalho;
  - Se houver um número de quadros extras suficiente, outro processo poderá ser iniciado.

Conjunto de Trabalho

- Manter uma janela do conjunto de trabalho inteira é custoso;
- Problema dessa abordagem é definir o controle da janela;
- Como isso ocorre em um SO:
- Ter uma “aproximação de uso”
  - Usa manter um intervalo fixo de tempo (interrupção) e um bit de referência (ref_bit);
  - O ref_bit é configurado com 1 quando a página é referenciada;
  - Exemplo:
    - Se assumir que Δ é igual a 10.000 referências;

Conjunto de Trabalho

- Exemplo: Δ = 10.000 referências a memória:
- Interrupção de tempo (WS_t) a cada 5.000 referências;
- Deve guarda na memória os bits para cada página:
  - Quando uma interrupção ocorrer (a cada 5000 ref.), deve copiar o ref_bit e reinicia o ref_bit de cada página;
  - Quando ocorre um page fault, verifica os bits (ref_bit) na memória;
  - Se qualquer um deles for igual 1;
  - a página foi usada no intervalo das ultimas 10.000 a 15.000 referências e coloca a página no conjunto de trabalho.

Conjunto de Trabalho

- Com WS, o mais importante é o seu tamanho, que pode ser medido pela demanda de quadros;

- Demanda de quadros pode ser dada por:
  - $D = \Sigma\ WSS_i \equiv$ total de quadros demandados;
  - $WSS_i \rightarrow$ tamanho do conjunto de trabalho de um processo P_i ;
  - Dado que m é o tamanho da memória, em quadros:
  - Se $D > m$ ocorre um thrashing.
- Política: Se $D >$ m, então suspende um dos processos:
  - Manter o WS é custoso.
- Mas existe alguma outra forma de controlar o thrashing?
  - Controle por meio de monitoria.

Conjunto de Trabalho

- O SO deve monitorar a taxa de page-fault e aumentar/decrementar alocação baseado em:
  - Selecione um intervalo aceitável de taxa page-fault;
  - Se a taxa atual ultrapassar o limite deve alocar outros quadros,
  - Se a taxa ficar abaixo, remove quadro que não são empregados.

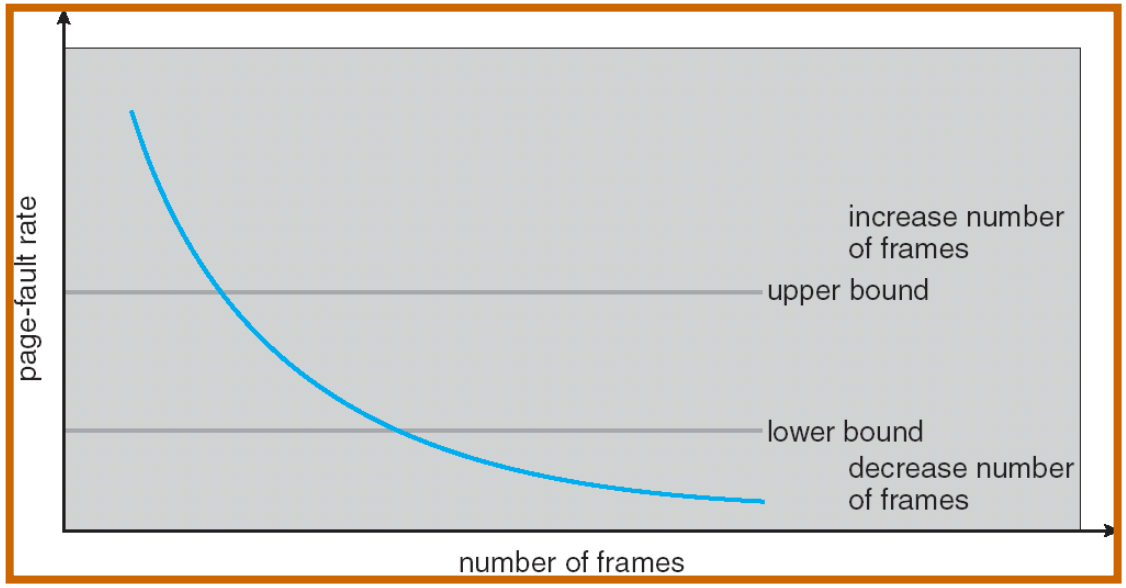


Figura 12. Estabelecer os intervalos de taxa de page-fault

Outras Considerações

- Impacto na seleção do tamanho da página
  - Fragmentação
  - Tamanho da tabela de página
  - Overhead E/S
- Pré-paginação
  - Trazer para memória de uma só vez todas as páginas necessárias:
    - Reduz o número de page faults nos processos iniciados
    - Mas muitas páginas trazidas para memória pode não serem usadas
    - Exemplo: Solaris (apenas para arquivos pequenos)

Outras Considerações: Estrutura de Programa

Estrutura de um programa

- Programa em Java
  - int data [128][128];
  - Cada linha é armazenada em uma página; quadros alocados <128
  - Quantas falhas de páginas acontecem em cada programa?
  - Programa 1

```
for (j = 0; j < 128; j++)
 for (i = 0; i < 128; i++)
     data[i][j] = 0;
```

#page faults: 128 x 128 = 16.384

Outras Considerações: Estrutura de Programa

Estrutura de um programa

- Programa em Java
- int data [128][128];
  - Programa 2
  -

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i][j] = 0;
```

#page faults: 128

  - Todas as palavras são zeradas antes de iniciar a próxima página.

Exemplo em SO:

Windows 10

- Em 32 bits, o espaço de endereço virtual é de 2 GB para um processo e suporta 4 GB de memória física.
- Em 64 bits, tem espaço de 128 TB do processo e 24 TB de memória física.
- Usa paginação por demanda com cluster: o cluster trata não só a página que falhou mas várias páginas posteriores.
- Processo é criado com um WS de mínimo de 50 páginas e máximo de 345 páginas;
- Usa uma variação do algoritmo do relógio com LRU com uma combinação de política de substituição local e global.
- Corte automático do WS: se memória livre no sistema falha permite um corte, remove páginas de um processo acima do mínimo do WS;

Exemplo em SO:

Linux:

- Linux usa paginação por demanda, alocando páginas de uma lista de quadros livres;
- Usa uma política de substituição de página global semelhante ao algoritmo do relógio com LRU;
- Existem 2 tipos de lista de páginas: lista ativa com páginas em uso; e lista inativa com páginas não recentemente usadas;

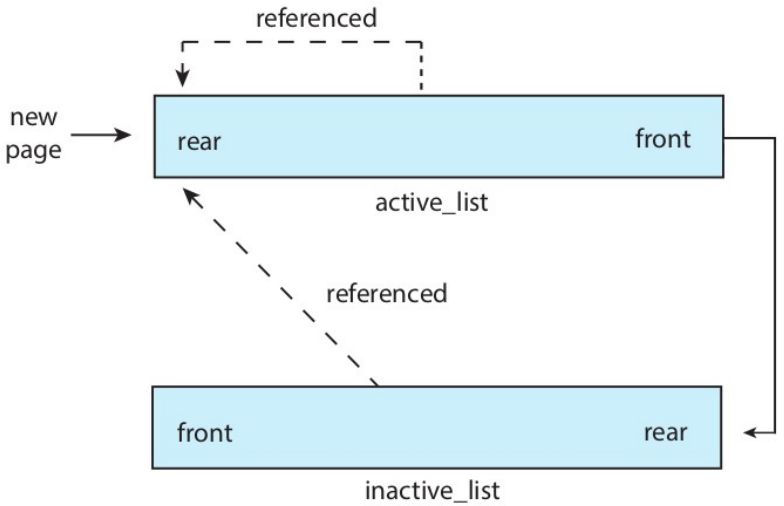


Figura 13. Estrutura de Listas da Memória

Exercício

Um sistema com gerência de memória virtual por paginação possui tamanho de página com 512 Bytes, espaço de endereçamento virtual com 512 Bytes páginas endereçadas de 0 à 511. A memória RAM tem 10 páginas numeradas de 0 à 9.

O conteúdo atual da memória RAM contém apenas informações de um processo conforme Tabela abaixo:

Endereço Físico	Conteúdo
1536	Página Virtual 34
2048	Página Virtual 9
3072	Tabela de páginas
3584	Página Virtual 65
4608	Página Virtual 10

a) Considere que a entrada da tabela de páginas contém, além do endereço do quadro, também o número da página. Mostre o conteúdo da tabela de páginas deste processo.

b) Mostre o conteúdo da tabela de páginas após a página virtual 9 ser carregada na memória a partir do endereço RAM 0 e a página virtual 34 ser substituída pela página virtual 12.

c) Qual endereço físico está associado ao endereço virtual 4613?

Exercício

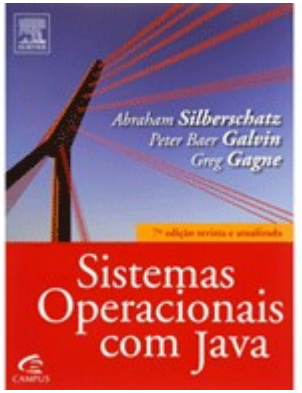
- Um computador tem 4 molduras de página. O tempo de carregamento de página na memória, o instante do último acesso e os bits R e M para cada página são mostrados a seguir.

Página	Carregado	Última Ref	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

- Qual página será trocada pelos algoritmos: segunda chance melhorado, FIFO, LRU e segunda chance?

Leituras Sugeridas

- Silberschatz, A., Galvin, P. B. Gagne, G. Sistemas Operacionais com Java. 7º edição. Editora Campus, 2008.



- TANENBAUM, A. Sistemas Operacionais Modernos. Rio de Janeiro: Pearson, 3 ed. 2010