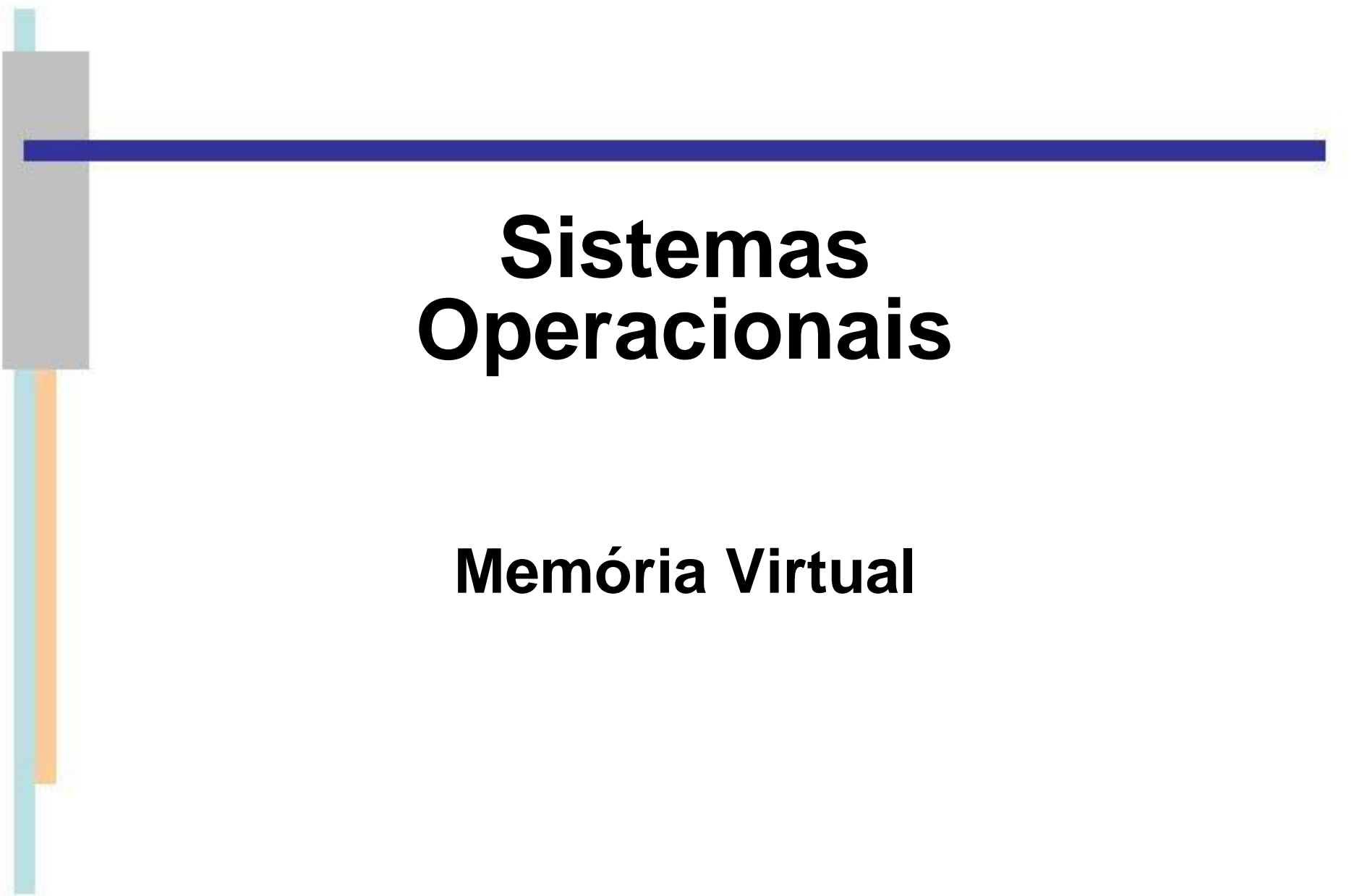


Sistemas Operacionais

Memória Virtual
Localidade de referências



Sistemas Operacionais

Memória Virtual

Problema

Grande problema dos computadores

→ disponibilidade de **memória física**

- *imagens, áudio, vídeo...* contribuem para esse problema
- informações volumosas
- exigem grandes quantidades de memória livre

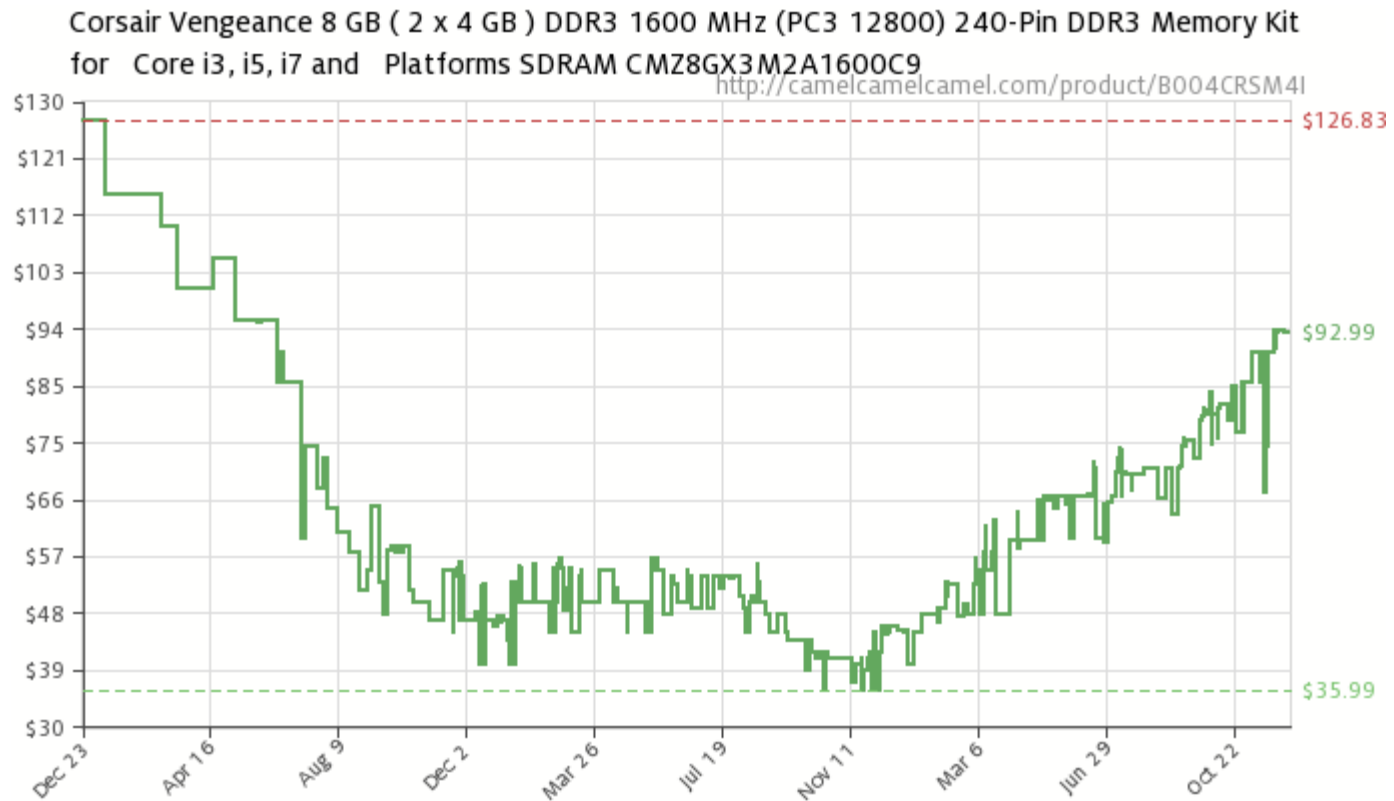
Solução (??)

- Há um custo associado

Ano	Custo Médio por Gigabyte
2013	\$5.5
2010	\$12.37
2005	\$189
2000	\$1,107
1995	\$30,875
1990	\$103,880
1985	\$859,375
1980	\$6,328,125

Solução (??)

- Há um custo associado



Solução (??)

- **Consome muita energia**
 - **Aumentar** sua capacidade nem sempre é uma opção factível

Fatos

Em um sistema computacional

- nem todos os processos estão constantemente ativos
- nem todas as áreas de memória de um processo ativo estão constantemente sendo usadas

Solução

Áreas de memória pouco acessadas:

transferidas para um meio de
armazenamento mais barato e
abundante

disco rígido
ou banco de memória *flash*

→ RAM liberada

Custo médio do HD

Ano	Custo Médio por Gigabyte
2013	\$0.05
2010	\$0.09
2005	\$1.24
2000	\$11.00
1995	\$1,120
1990	\$11,200
1985	\$105,000
1980	\$437,500

Memória Virtual



O uso de um **armazenamento externo** como extensão da memória RAM se chama *memória virtual*

Memória Virtual – Mecanismo básico

Swapping

Nos *primeiros sistemas*, processos inteiros eram **transferidos** da memória para o disco rígido e vice-versa

Só se justifica nos casos em que o local de armazenamento é muito lento.

Memória Virtual – Mecanismo básico

Paging

Nos *sistemas atuais* as transferências são feitas por páginas ou grupos de páginas

Memória Virtual – Mecanismo básico

O mecanismo de memória virtual se baseia em **páginas** em vez de **segmentos**, pois...

Tamanho fixo da página – permite:

- simplificar os algoritmos de escolha de páginas a remover
- simplificar os mecanismos de transferência para o disco
- simplificar a formatação da **área de troca**

Área de Troca

O armazenamento externo pode ser feito:

- **Em um disco exclusivo** (servidores de maior porte);

Área de Troca

O armazenamento externo pode ser feito:

- Em um disco exclusivo (servidores de maior porte);
- Em uma partição do disco principal (Linux e UNIX);

Área de Troca

O armazenamento externo pode ser feito:

- Em um disco exclusivo (servidores de maior porte);
- Em uma partição do disco principal (Linux e UNIX);
- Em um arquivo reservado, geralmente oculto (Windows NT e sucessores);

Área de Troca

O armazenamento externo pode ser feito:

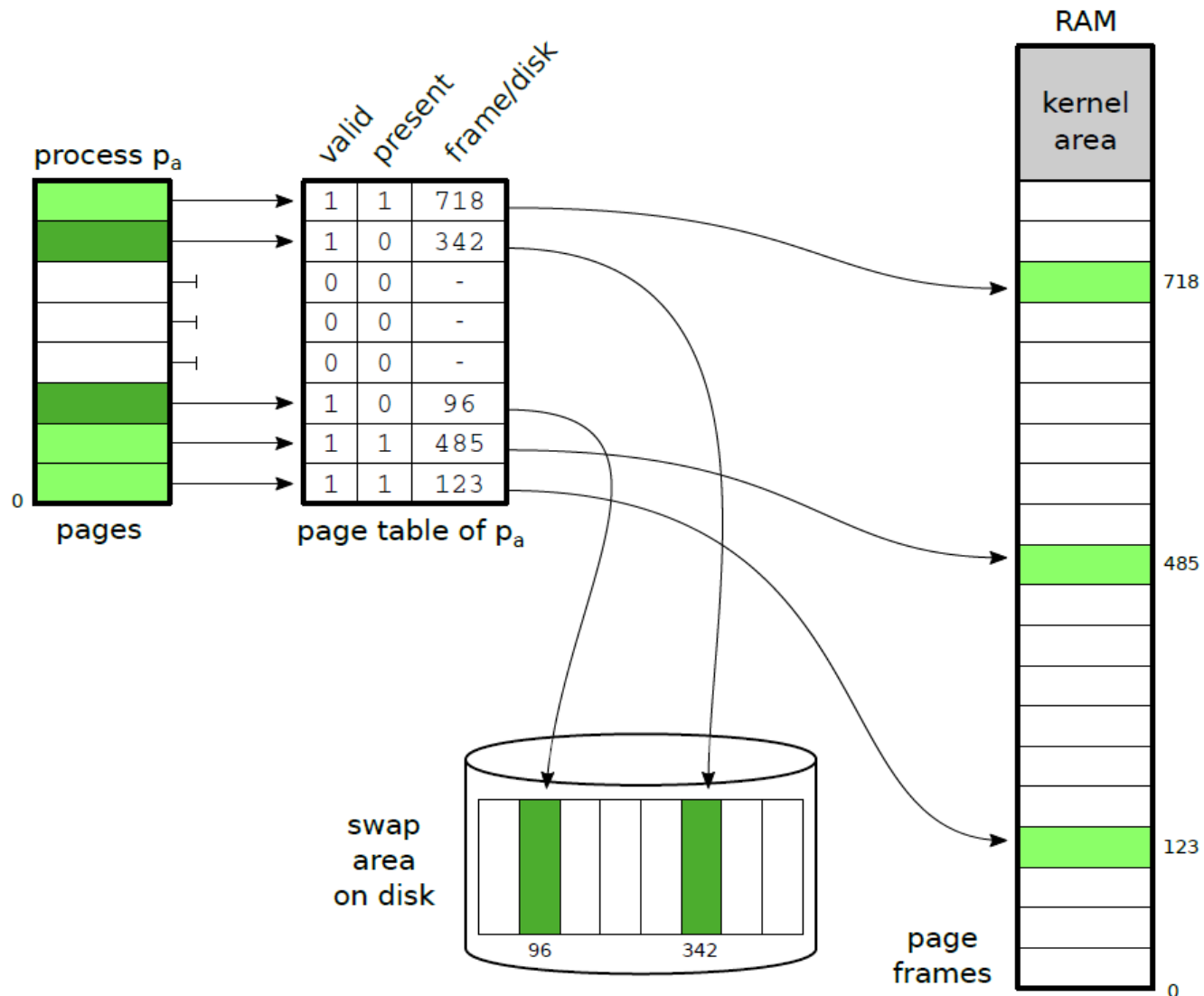
- Em um disco exclusivo (servidores de maior porte);
- Em uma partição do disco principal (Linux e UNIX);
- Em um arquivo reservado, geralmente oculto (Windows NT e sucessores);
- Em um servidor de arquivos de rede (apresenta baixo desempenho).

Idéia Central

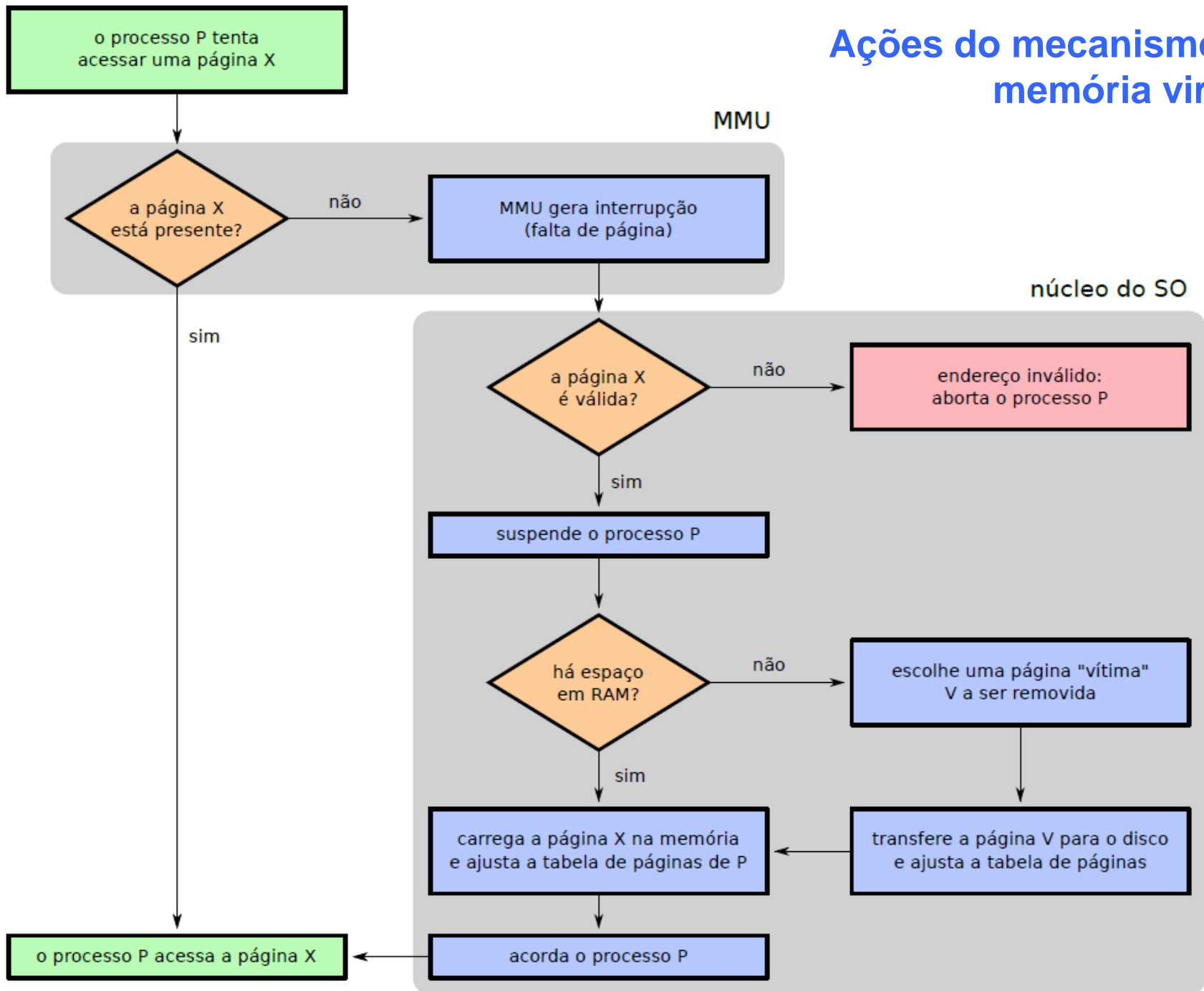
Memória paginada

- Retirar da memória principal as **páginas pouco usadas** dos processos
- **⇒ Tabelas de páginas** relativas aos quadros transferidos são ajustadas

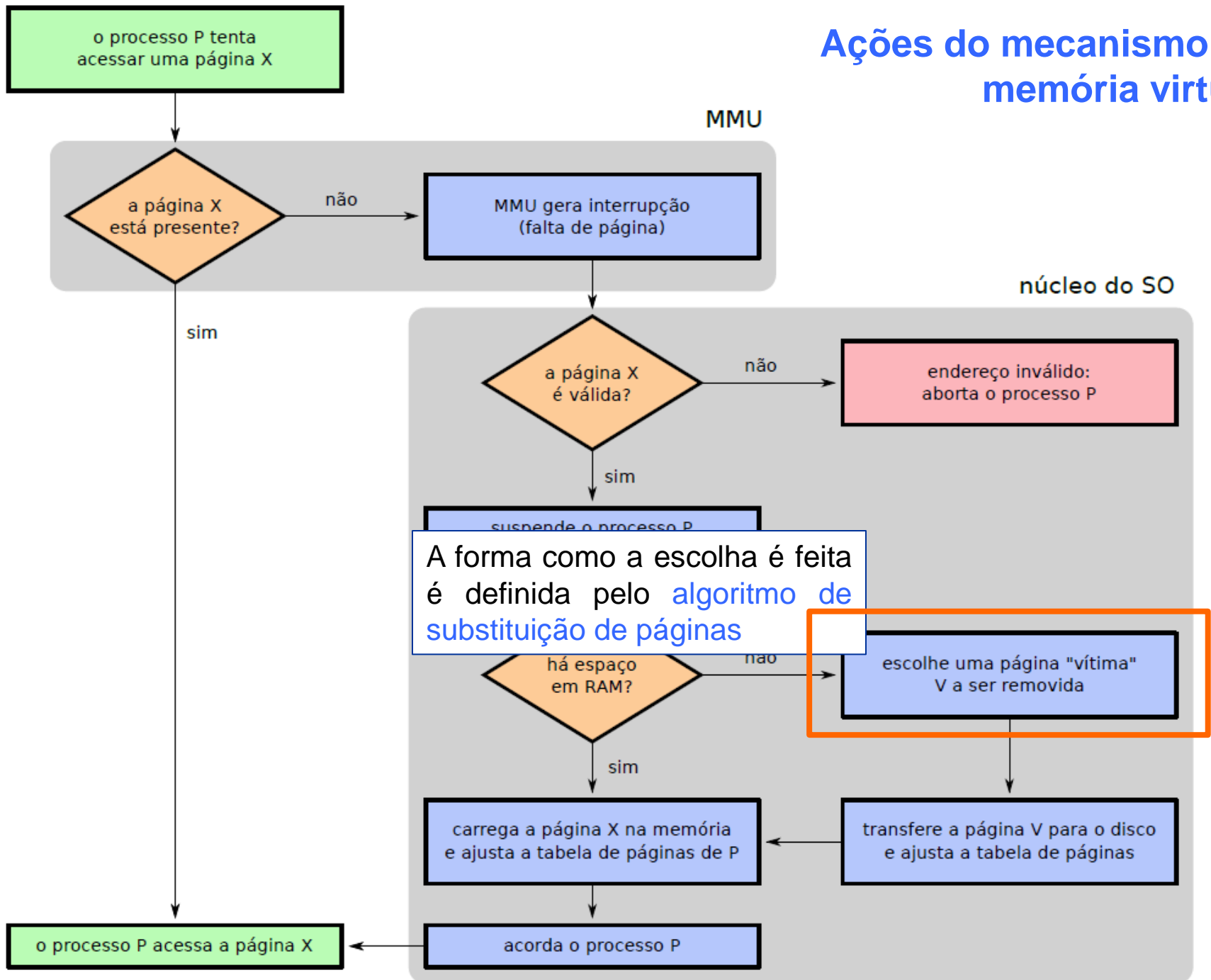
Memória Virtual Paginada



Ações do mecanismo de memória virtual



Ações do mecanismo de memória virtual



Questões importantes

Memória principal cheia

Uma ou mais páginas deverão ser removidas

- Alguns sistemas, como o Linux e o Solaris, mantêm um processo *daemon* com a finalidade de escolher e transferir páginas para o disco
- Ativado sempre que a quantidade de memória livre estiver abaixo de um limite mínimo

Questões importantes

Retomar a execução do processo pode ser uma tarefa complexa

Instruções que envolvam **várias ações** e **vários endereços** de memória → qual endereço gerou falta de página?

Questões importantes

Retomar a execução do processo pode ser uma tarefa complexa

Instruções que envolvam **várias ações** e **vários endereços** de memória → qual endereço gerou falta de página?

A maioria dos processadores atuais provê registradores especiais que auxiliam nessa tarefa

Memória virtual → depende do processador

Eficiência de Uso

Disco visto como uma extensão da RAM de forma **transparente** para os processos.

Porém:

Disco é cerca de **100.000** vezes mais lento que a RAM!

Eficiência de Uso

Como calcular?

R_ Conceito de tempo médio ponderado

Eficiência de Uso

Como calcular?

R_ Conceito de tempo médio ponderado

$$t_m = (1 - p_a)t_e + p_at_a$$

Eficiência de Uso

Como calcular?

Exemplo:

- Tempo de acesso da RAM = 60 ns;
- Tempo de acesso do Disco de troca = 6 ms;
- 1 falta de página a cada 10^6 acessos;

Eficiência de Uso

1 – Memória NÃO saturada:

$$t_{medio} = \frac{(999.999 \times 60 \text{ ns}) + 1 \times 6 \text{ ms}}{1.000.000} = 66 \text{ ns}$$

$\Rightarrow 10\%$

Eficiência de Uso

1 – Memória saturada:

$$t_{medio} = \frac{(999.999 \times 60 \text{ ns}) + 2 \times 6 \text{ ms}}{1.000.000} = 72 \text{ ns}$$

$\Rightarrow 20\%$

Eficiência de Uso

1 – Memória saturada:

$$t_{medio} = \frac{(999.999 \times 60 \text{ ns}) + 2 \times 6 \text{ ms}}{1.000.000} = 72 \text{ ns}$$


⇒ 20%

Caso a frequência de falta de páginas aumente para uma falta a cada 100.000 acessos, o tempo médio de acesso à memória subirá para 180 ns → 3 vezes mais lento.

Eficiência de Uso

A frequência de faltas de página depende:

1. Do comportamento dos processos
 - localidade de referências
 - agrupam acessos em poucas páginas
2. Escolha das páginas a remover
 - algoritmos de substituição
3. Do tamanho da memória RAM
 - fenômeno de *thrashing*



Sistemas Operacionais

Localidade de Referências

Localidade de referências

- A forma como os processos acessam a memória tem um impacto direto na eficiência dos mecanismos de gerência de memória
 - sobretudo o cache de páginas (TLB, visto anteriormente)
 - e o mecanismo de memória virtual (será visto a seguir)

Localidade de referências

- Processos que **concentram** seus acessos em poucas páginas de cada vez farão um uso eficiente desses mecanismos
- Processos que acessam muitas páginas **distintas** em um curto período irão gerar frequentes erros de cache (TLB) e faltas de página

Localidade de referências

A propriedade de um processo ou sistema concentrar seus acessos em poucas áreas da memória a cada instante é chamada ***localidade de referências***

Localidade de referências

Localidade temporal: um recurso usado há **pouco tempo** será provavelmente usado novamente em um futuro próximo

Localidade de referências

Localidade temporal: um recurso usado há **pouco tempo** será provavelmente usado novamente em um futuro próximo

Localidade espacial: um recurso será mais provavelmente acessado se outro recurso **próximo** a ele já foi acessado

Localidade de referências

Localidade temporal: um recurso usado há **pouco tempo** será provavelmente usado novamente em um futuro próximo

Localidade espacial: um recurso será mais provavelmente acessado se outro recurso **próximo** a ele já foi acessado

Localidade sequencial: é um caso particular da localidade espacial no qual há uma predominância de **acesso sequencial** aos recursos

Importância da codificação

Exemplo com matriz

Considere um programa para o preenchimento de uma matriz de 4.096×4.096 bytes, onde **cada linha da matriz está alocada em uma página distinta**.

1. Percorrendo a matriz linha por linha:

```
unsigned char buffer[4096][4096] ;

int main ()
{
    int i, j ;

    for (i=0; i<4096; i++) // percorre as linhas do buffer
        for (j=0; j<4096; j++) // percorre as colunas do buffer
            buffer[i][j]= (i+j) % 256 ;
}
```


Exemplo com matriz

2. Percorrendo a matriz coluna por coluna:

```
unsigned char buffer[4096][4096] ;

int main ()
{
    int i, j ;

    for (j=0; j<4096; j++) // percorre as colunas do buffer
        for (i=0; i<4096; i++) // percorre as linhas do buffer
            buffer[i][j]= (i+j) % 256 ;
}
```

Localidade de referências

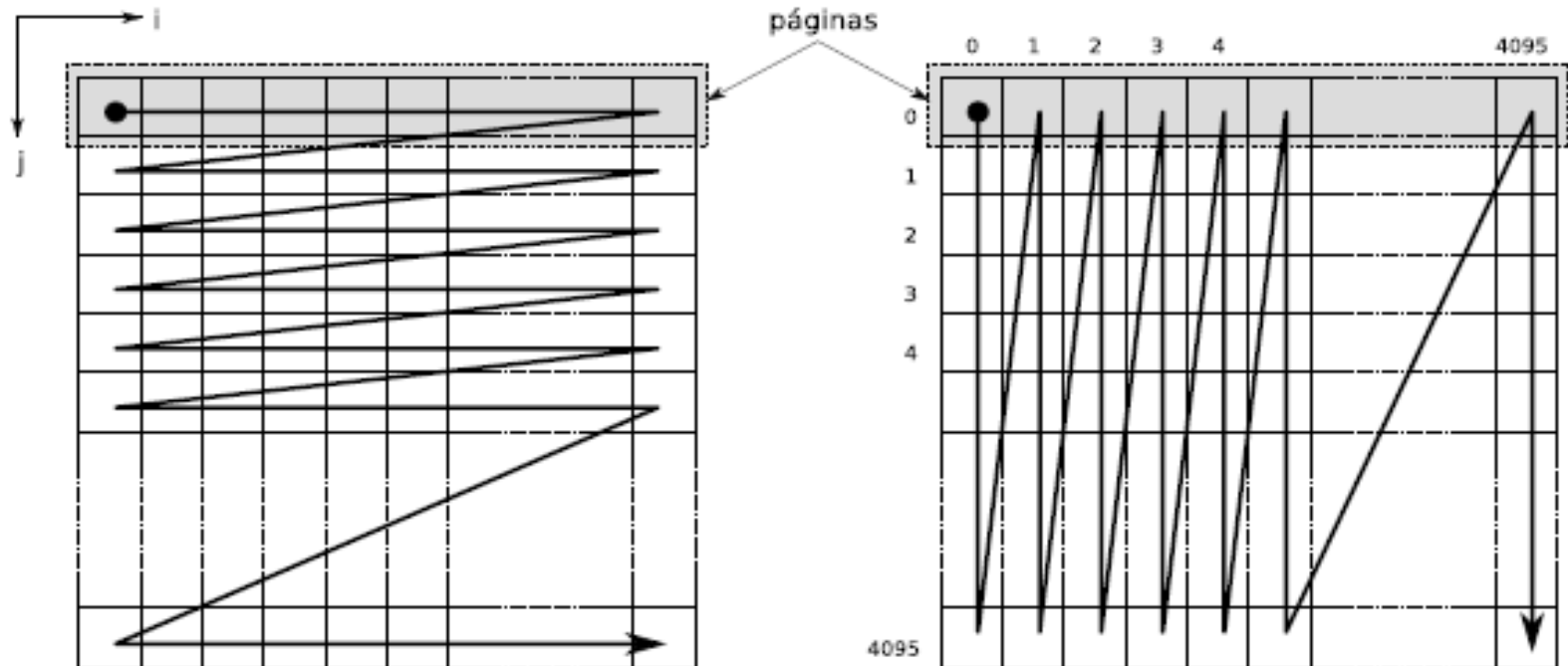
- Os dois programas geram o mesmo resultado e são conceitualmente equivalentes
- Entretanto, eles **não** têm o mesmo desempenho

Localidade de referências

- Os dois programas geram o mesmo resultado e são conceitualmente equivalentes
- Entretanto, eles **não** têm o mesmo desempenho
- A primeira implementação usa de forma eficiente o cache da tabela de páginas
- A implementação com percurso por colunas gera um erro de cache TLB a cada célula acessada

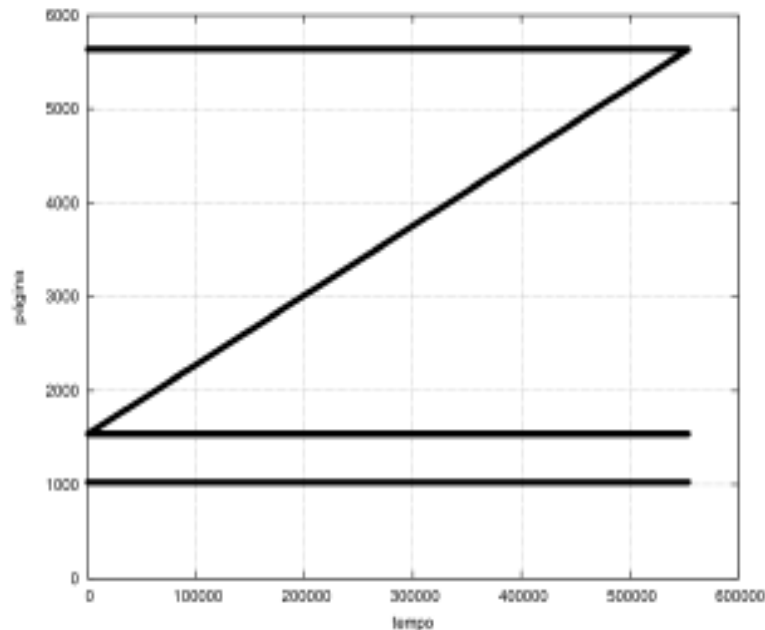
Localidade de referências

Comportamento dos programas no acesso à memória

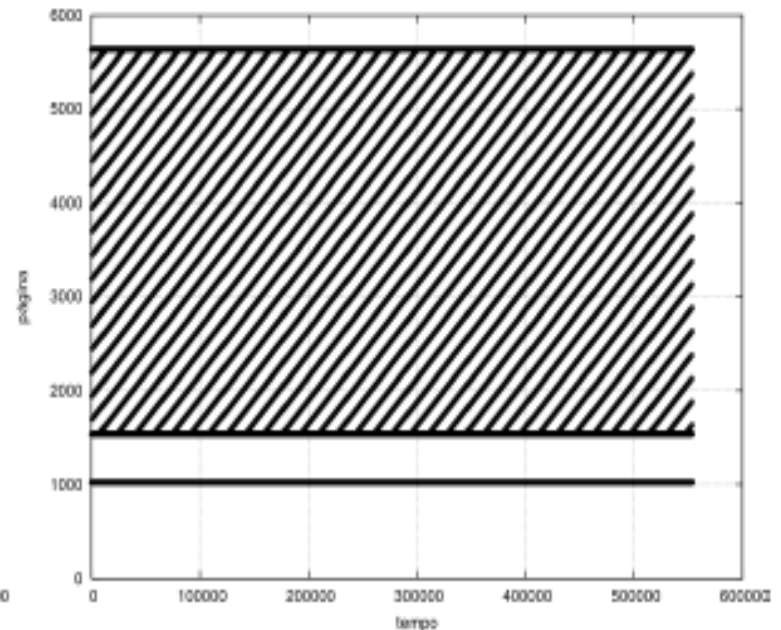


Localidade de referências

Localidade de referências nas duas execuções (**Mapa de acesso**)



Média: 5 páginas distintas em cada 100.000 acessos à memória



Média: 3.031 páginas distintas em cada 100.000 acessos

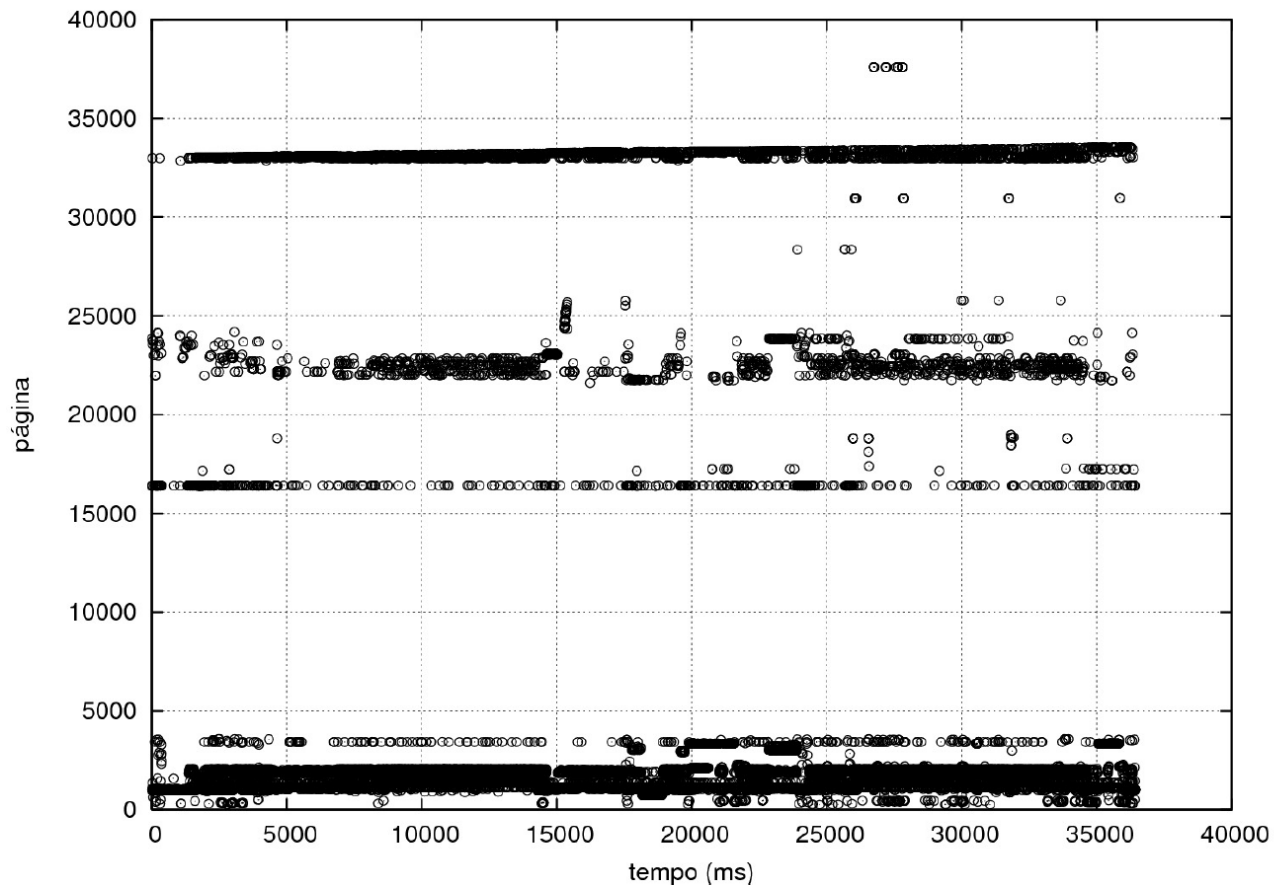
Localidade de referências

A **localidade de referência** de uma implementação depende de um conjunto de fatores:

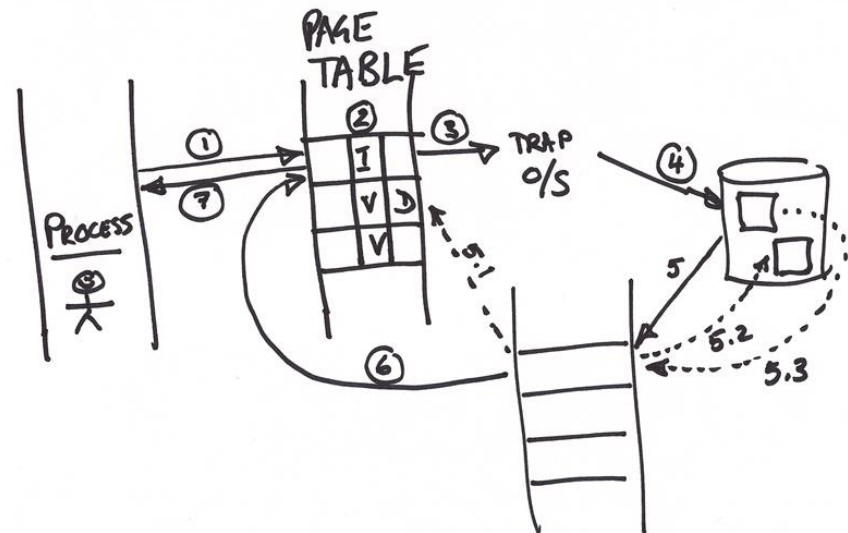
- As estruturas de dados usadas pelo programa;
 - vetores, matrizes, listas encadeadas e árvores, etc.
- Os algoritmos usados pelo programa
- A qualidade do compilador

Localidade de referências

Exemplo de aplicativo real (*gThumb*):



Algoritmos de Substituição de Páginas



Algoritmos de substituição de páginas

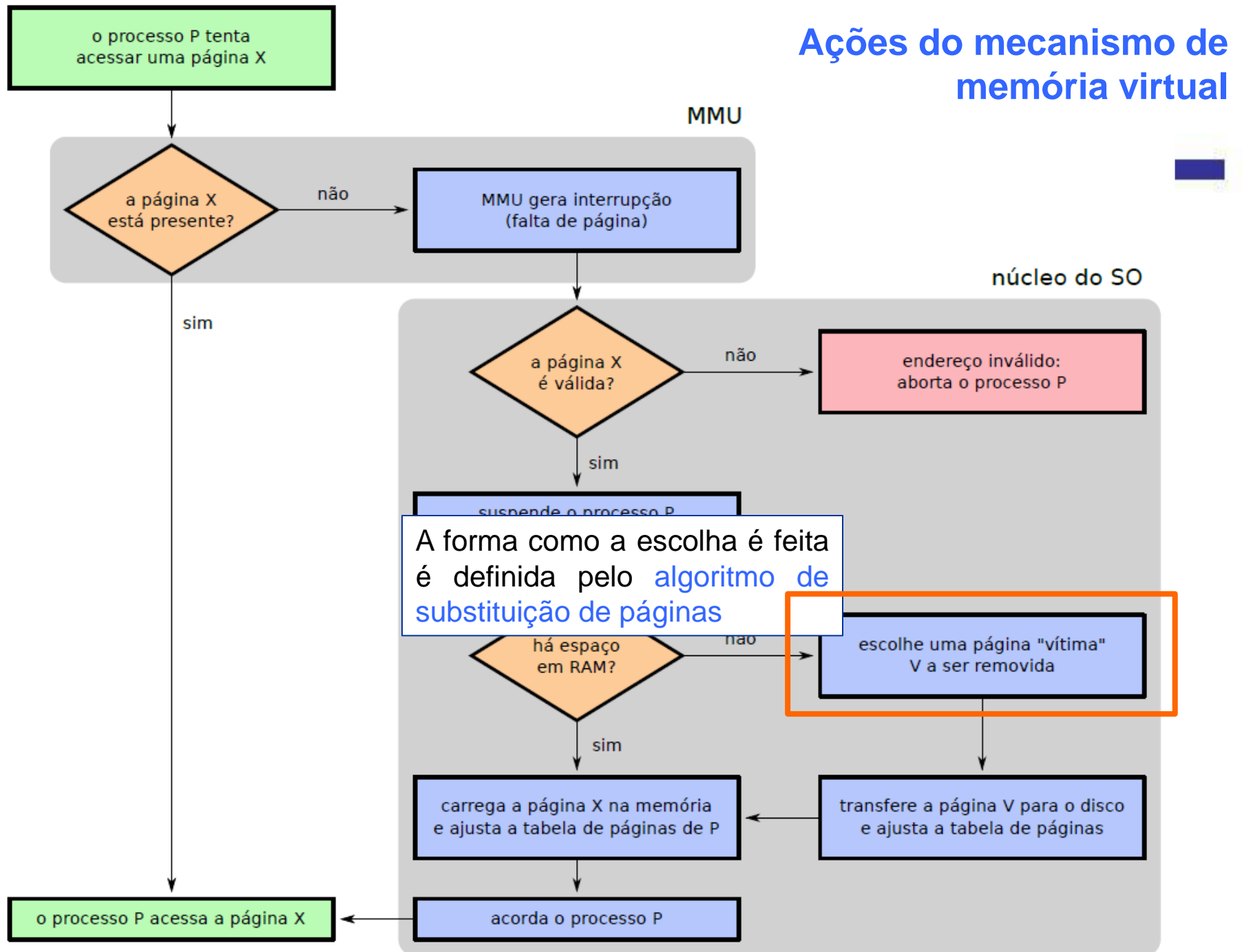
Premissa:

Remoção de páginas usadas com muita frequência...

...**aumenta** taxa de falta de página

...**diminui** o desempenho do sistema

Ações do mecanismo de memória virtual



Algoritmos de substituição de páginas

Crítérios

1. Idade da página
2. Frequência de acessos à página
3. Data do último acesso
4. Prioridade do processo proprietário
5. Conteúdo da página
6. Páginas especiais

Algoritmos de substituição de páginas

Critérios

1 - Idade da página

Há **quanto tempo** a página está na memória?

Páginas muito **antigas** talvez sejam **pouco usadas** num futuro próximo

Algoritmos de substituição de páginas

Critérios

2 - Frequência de acessos à página

Páginas acessadas em um **passado recente** possivelmente ainda o serão em um **futuro próximo**

Algoritmos de substituição de páginas

Critérios

3 - Data do último acesso

Páginas há muito tempo sem acesso possivelmente serão pouco acessadas em um futuro próximo

Algoritmos de substituição de páginas

Critérios

4 - Prioridade do processo proprietário

Processos de **alta prioridade**, ou de tempo-real, podem precisar de suas páginas de memória **rapidamente**

Se elas estiverem **no disco**, seu **desempenho** e **tempo de resposta** poderão ser prejudicados

Algoritmos de substituição de páginas

Critérios

5 - Conteúdo da página

Páginas cujo **conteúdo** seja código executável exigem menos esforço do mecanismo de memória virtual

Porque seu conteúdo **já está mapeado** no disco

Algoritmos de substituição de páginas

Critérios

5 - Conteúdo da página

Por outro lado, páginas de **dados** que foram **alteradas** precisam ser gravadas na área de troca

Algoritmos de substituição de páginas

Critérios

6 - Páginas especiais

Páginas contendo *buffers de E/S* podem levar a *perda* caso não estejam na memória no momento da transferência de dados entre o processo e o periférico

Algoritmos de substituição de páginas

Critérios

6 - Páginas especiais

Páginas contendo *buffers de E/S* podem levar a *perda* caso não estejam na memória no momento da transferência de dados entre o processo e o periférico

Páginas contendo *informações sensíveis* (como senhas) não devem ser copiadas na área de troca, por *segurança*

Algoritmos de substituição de páginas

Algoritmos que visam reduzir a frequência de falta de página:

- Algoritmo ÓTIMO
- FIFO
- LRU
- Algoritmo da SEGUNDA CHANCE
- NRU
- Envelhecimento
- Working Set

Cadeia de referências

Indica a sequência de páginas acessadas por um processo durante sua execução

Ex.:

0, 2, 1, 3, 5, 4, 6, 3, 7, 4, 7, 3, 3, 5, 5, 3, 1, 1, 1, 7, 1, 3, 4, 1

Cadeia de referências

Cadeias de referências de execuções **reais** podem ser muito longas:

$$T_s = 50 \text{ ns}$$

Em um segundo → 20 milhões de acesso!

Cadeia de referências

Pode-se inferir a respeito da **eficiência de um algoritmo** submetendo-se a ele a **cadeia de referências** de uma execução

Cadeia de referências

- Obtenção de cadeias de referências confiáveis é uma área de pesquisa importante
- Envolve técnicas complexas de coleta, filtragem e compressão de dados de execução de sistemas

A decorative graphic on the left side of the slide. It consists of a vertical gray rectangle. To its left, there are two thin vertical bars, one light blue and one orange. A thick dark blue horizontal line extends from the gray rectangle across the top of the slide.

Algoritmo ÓTIMO

1. Algoritmo ÓTIMO

Premissa:

A melhor página ser substituída é que possui menos chances de ser utilizada no futuro

1. Algoritmo ÓTIMO

- A melhor página a remover da memória em um dado instante é aquela que ficará **mais tempo sem ser usada** pelos processos

1. Algoritmo ÓTIMO

- A melhor página a remover da memória em um dado instante é aquela que ficará **mais tempo sem ser usada** pelos processos
- PORÉM, o comportamento **futuro** dos processos não pode ser previsto:
 - Algoritmo **não é implementável**
 - CONTUDO, é importante porque define um **limite mínimo** conceitual

0, 2, 1, 3, 5, 4, 6, 3, 7, 4, 7, 3, 3, 5, 5, 3, 1, 1, 1, 7, 1, 3, 4, 1

t	página	q_0	q_1	q_2	faltas	ação
1	0	0			★	p_0 é carregada no quadro vazio q_0
2	2	0	2		★	p_2 é carregada em q_1
3	1	0	2	1	★	p_1 é carregada em q_2
4	3	3	2	1	★	p_3 substitui p_0 (não será mais acessada)
5	5	3	5	1	★	p_5 substitui p_2 (não será mais acessada)
6	4	3	5	4	★	p_4 substitui p_1 (só será acessada em $t = 17$)
7	6	3	6	4	★	p_6 substitui p_5 (só será acessada em $t = 14$)
8	3	3	6	4		p_3 está na memória
9	7	3	7	4	★	p_7 substitui p_6 (não será mais acessada)
10	4	3	7	4		p_4 está na memória
11	7	3	7	4		p_7 está na memória
12	3	3	7	4		p_3 está na memória
13	3	3	7	4		p_3 está na memória
14	5	3	7	5	★	p_5 substitui p_4 (só será acessada em $t = 23$)
15	5	3	7	5		p_5 está na memória
16	3	3	7	5		p_3 está na memória
17	1	3	7	1	★	p_1 substitui p_5 (não será mais acessada)
18	1	3	7	1		p_1 está na memória
19	1	3	7	1		p_1 está na memória
20	7	3	7	1		p_7 está na memória
21	1	3	7	1		p_1 está na memória
22	3	3	7	1		p_3 está na memória
23	4	4	7	1	★	p_4 substitui p_3 (não será mais acessada)
24	1	4	7	1		p_1 está na memória

11 Faltas

2. FIFO

A escolha das páginas a substituir é sua
“idade”

0, 2, 1, 3, 5, 4, 6, 3, 7, 4, 7, 3, 3, 5, 5, 3, 1, 1, 1, 7, 1, 3, 4, 1

t	página	q_0	q_1	q_2	faltas	ação
1	0	0			★	p_0 é carregada no quadro vazio q_0
2	2	0	2		★	p_2 é carregada em q_1
3	1	0	2	1	★	p_1 é carregada em q_2
4	3	3	2	1	★	p_3 substitui p_0 (a mais antiga na memória)
5	5	3	5	1	★	p_5 substitui p_2 (idem)
6	4	3	5	4	★	p_4 substitui p_1
7	6	6	5	4	★	p_6 substitui p_3
8	3	6	3	4	★	p_3 substitui p_5
9	7	6	3	7	★	p_7 substitui p_4
10	4	4	3	7	★	p_4 substitui p_6
11	7	4	3	7		p_7 está na memória
12	3	4	3	7		p_3 está na memória
13	3	4	3	7		p_3 está na memória
14	5	4	5	7	★	p_5 substitui p_3
15	5	4	5	7		p_5 está na memória
16	3	4	5	3	★	p_3 substitui p_7
17	1	1	5	3	★	p_1 substitui p_4
18	1	1	5	3		p_1 está na memória
19	1	1	5	3		p_1 está na memória
20	7	1	7	3	★	p_7 substitui p_5
21	1	1	7	3		p_1 está na memória
22	3	1	7	3		p_3 está na memória
23	4	1	7	4	★	p_4 substitui p_3
24	1	1	7	4		p_1 está na memória

15 Faltas

2. FIFO

Apesar de ter uma implementação **simples**, na prática este algoritmo **não oferece bons resultados**

2. FIFO

Seu principal defeito é considerar somente a idade da página, sem levar em conta sua importância

2. FIFO

Seu principal defeito é considerar somente a idade da página, sem levar em conta sua importância

Páginas carregadas na memória há muito tempo podem ter sido frequentemente acessadas

3. LRU – *Least Recently Used*

A escolha recai sobre as páginas que estão na memória há mais tempo **sem ser acessadas**

⇒ Páginas antigas, mas de uso frequente **não são penalizadas**

0, 2, 1, 3, 5, 4, 6, 3, 7, 4, 7, 3, 3, 5, 5, 3, 1, 1, 1, 7, 1, 3, 4, 1

t	página	q_0	q_1	q_2	faltas	ação
1	0	0			★	p_0 é carregada no quadro vazio q_0
2	2	0	2		★	p_2 é carregada em q_1
3	1	0	2	1	★	p_1 é carregada em q_2
4	3	3	2	1	★	p_3 substitui p_0 (há mais tempo sem acessos)
5	5	3	5	1	★	p_5 substitui p_2 (idem)
6	4	3	5	4	★	p_4 substitui p_1 (...)
7	6	6	5	4	★	p_6 substitui p_3
8	3	6	3	4	★	p_3 substitui p_5
9	7	6	3	7	★	p_7 substitui p_4
10	4	4	3	7	★	p_4 substitui p_6
11	7	4	3	7		p_7 está na memória
12	3	4	3	7		p_3 está na memória
13	3	4	3	7		p_3 está na memória
14	5	5	3	7	★	p_5 substitui p_4
15	5	5	3	7		p_5 está na memória
16	3	5	3	7		p_3 está na memória
17	1	5	3	1	★	p_1 substitui p_7
18	1	5	3	1		p_1 está na memória
19	1	5	3	1		p_1 está na memória
20	7	7	3	1	★	p_7 substitui p_5
21	1	7	3	1		p_1 está na memória
22	3	7	3	1		p_3 está na memória
23	4	4	3	1	★	p_4 substitui p_7
24	1	4	3	1		p_1 está na memória

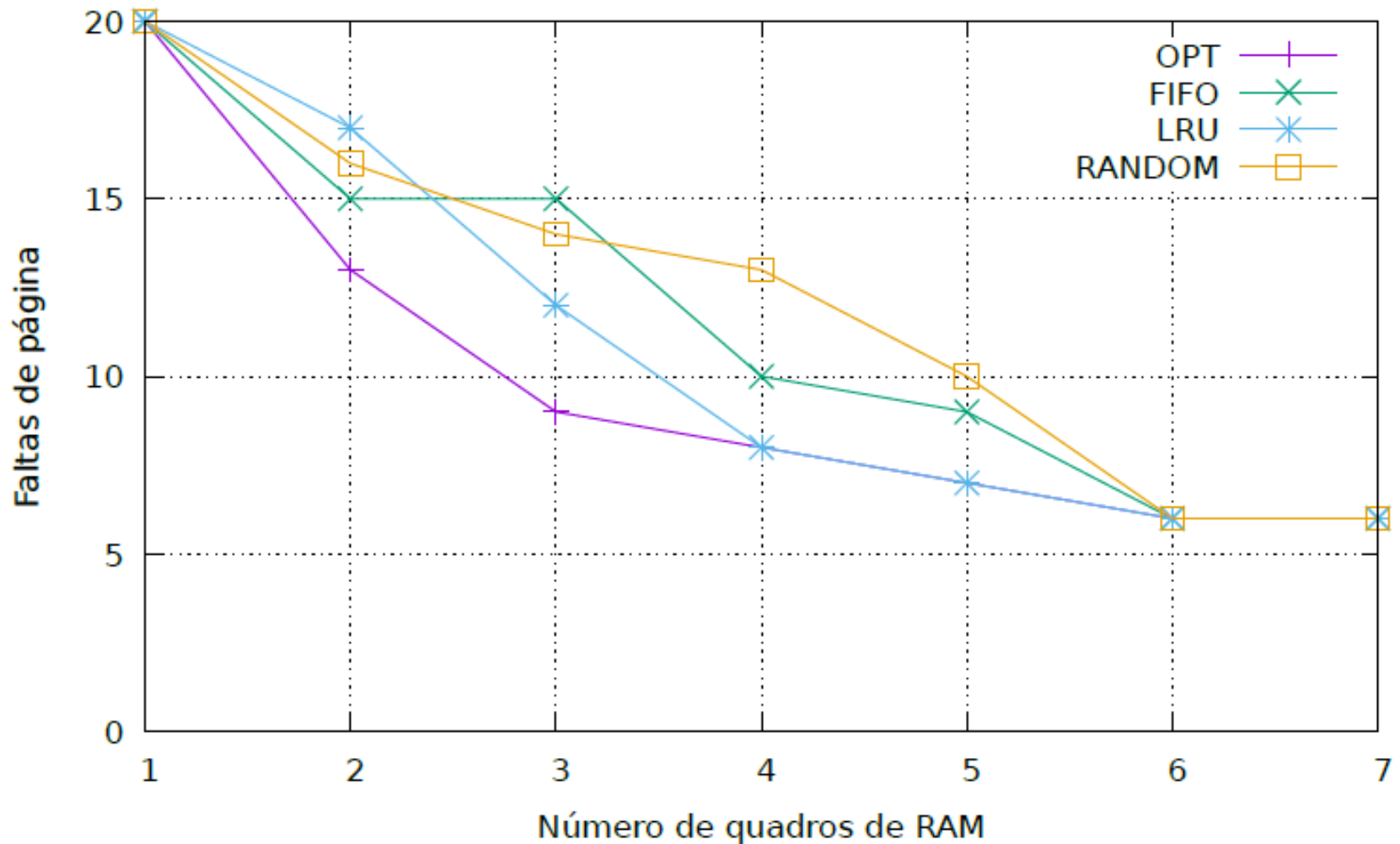
14 Faltas

3. LRU – Least Recently Used

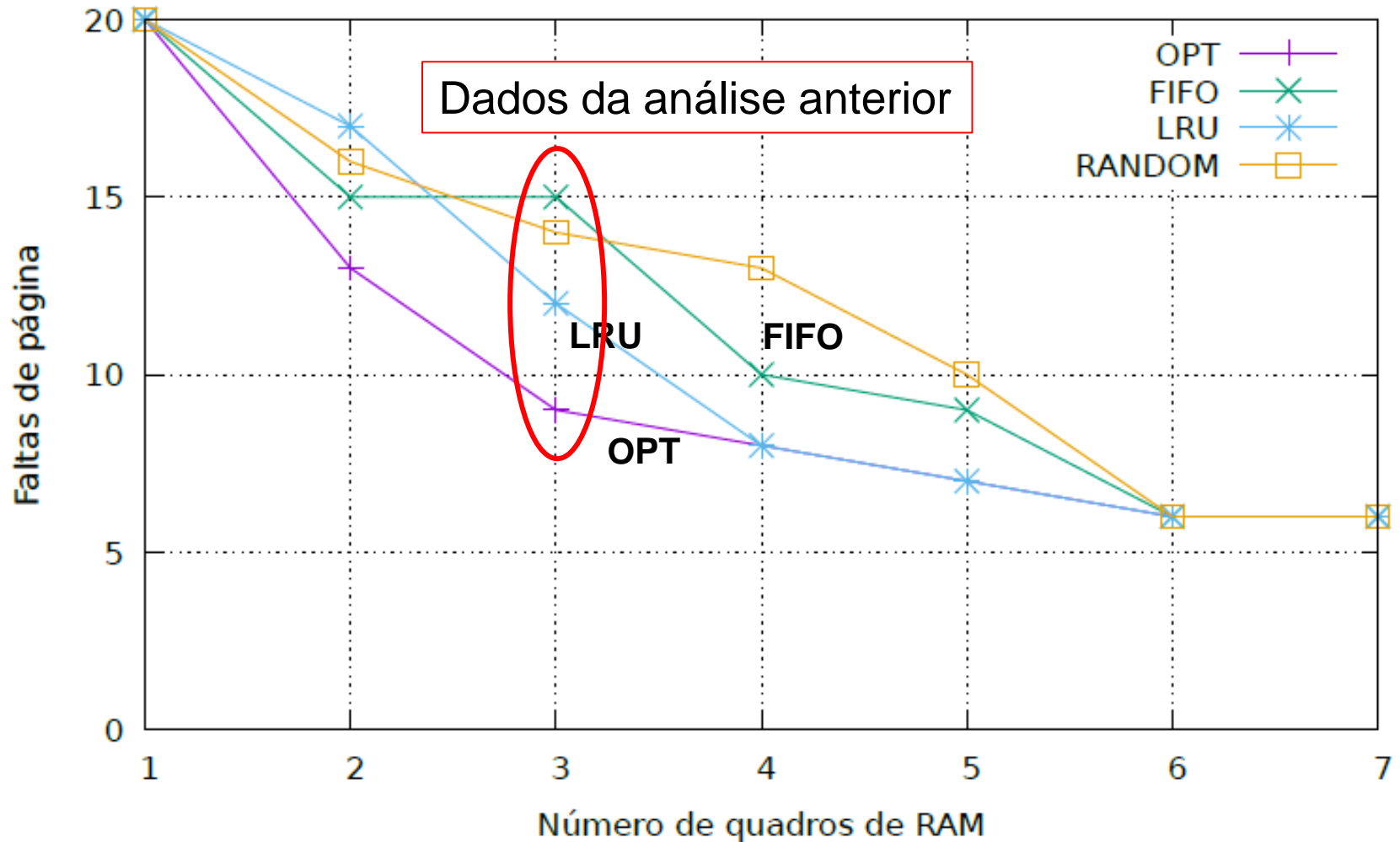
Este algoritmo é **simétrico** do algoritmo **Ótimo** em relação ao tempo:

- O **ótimo** busca as páginas que serão acessadas “**mais longe**” no **futuro** do processo
- O algoritmo LRU busca as páginas que foram acessadas “**mais longe**” no seu **passado**

Comparação entre algoritmos

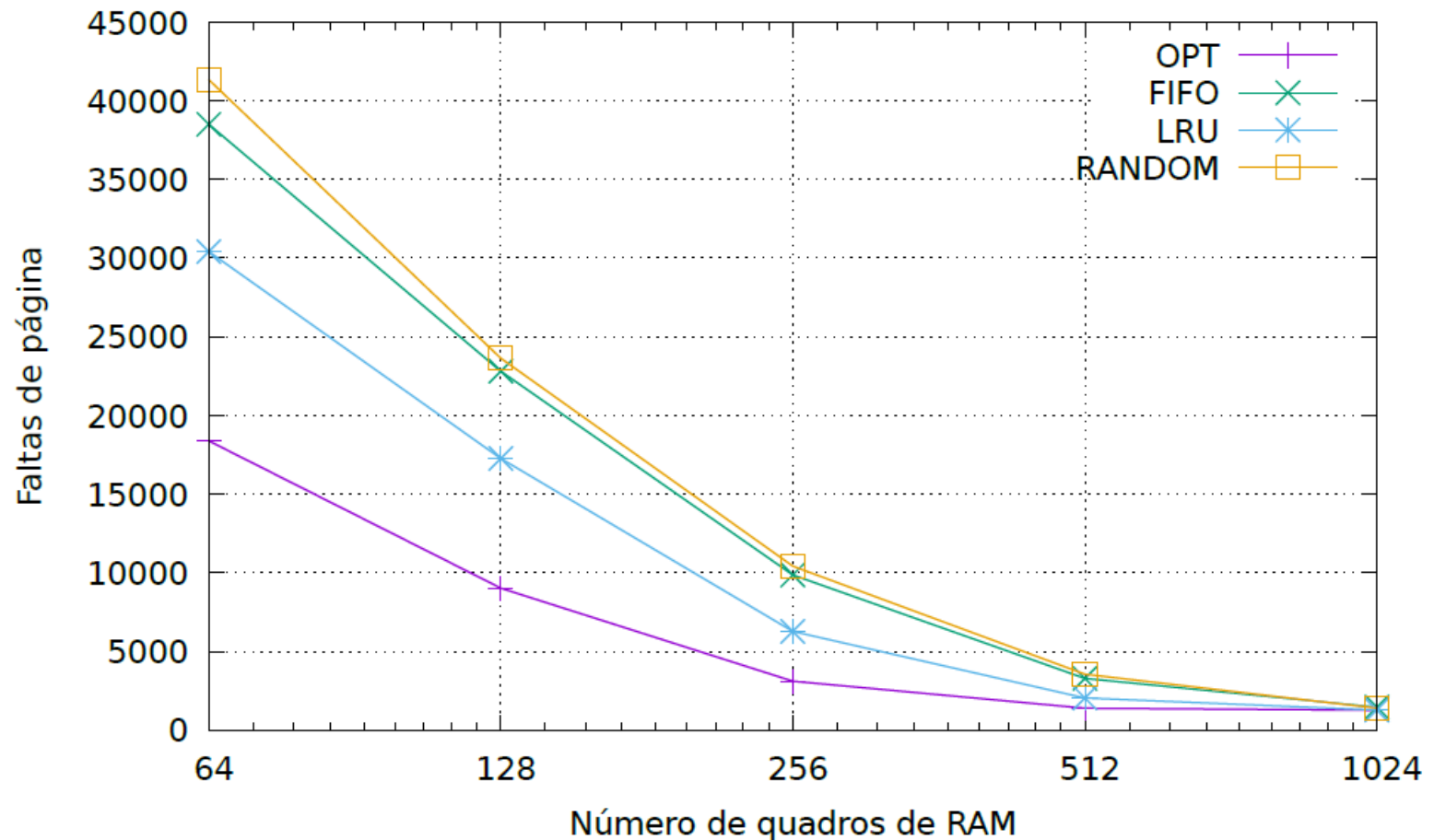


Comparação entre algoritmos



Comparação entre algoritmos

Cadeia de referências do compilador GCC (<http://gcc.gnu.org>) com 10^6 referências a 1.260 páginas distintas



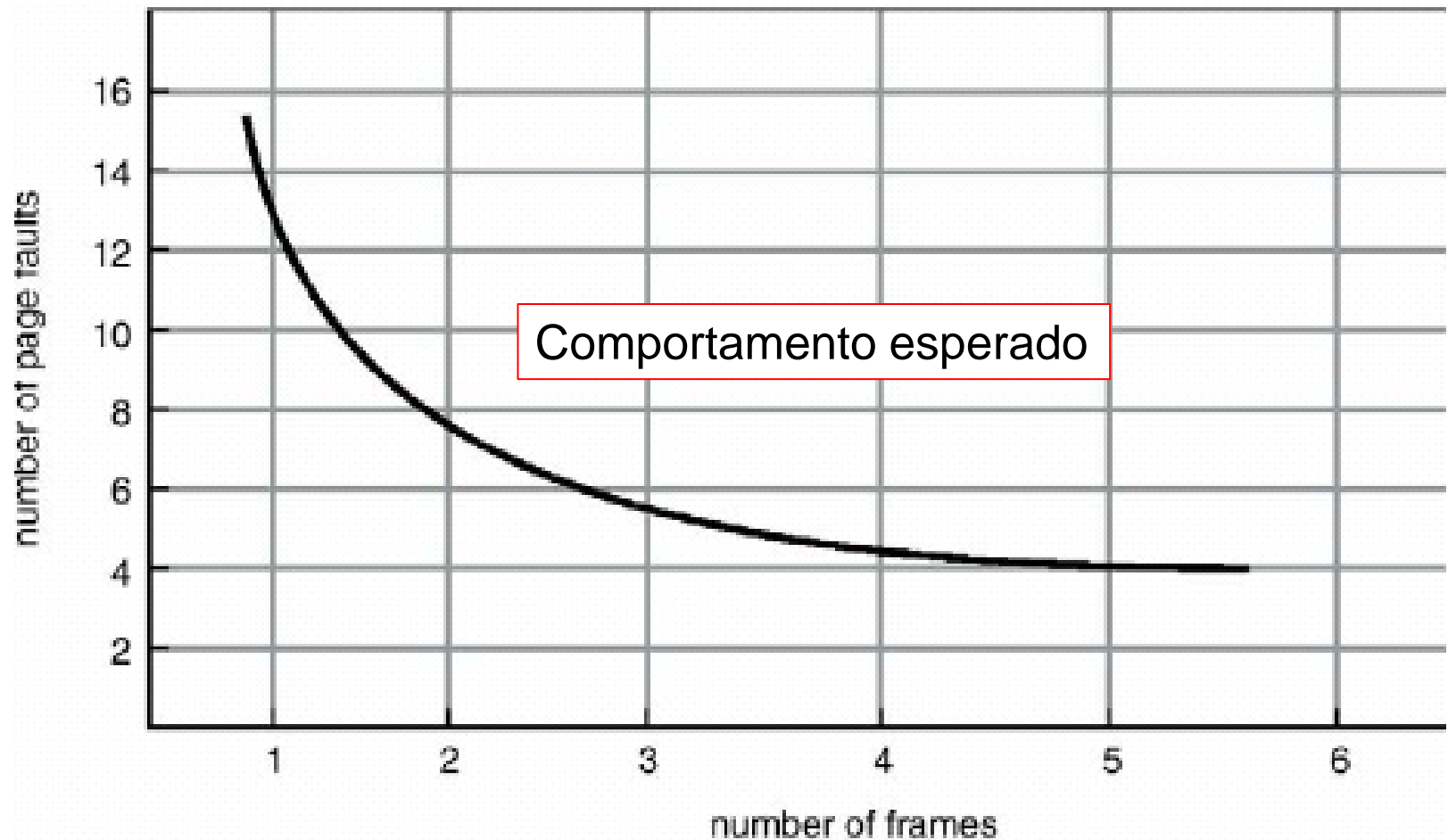


Anomalia de Belady (1969)

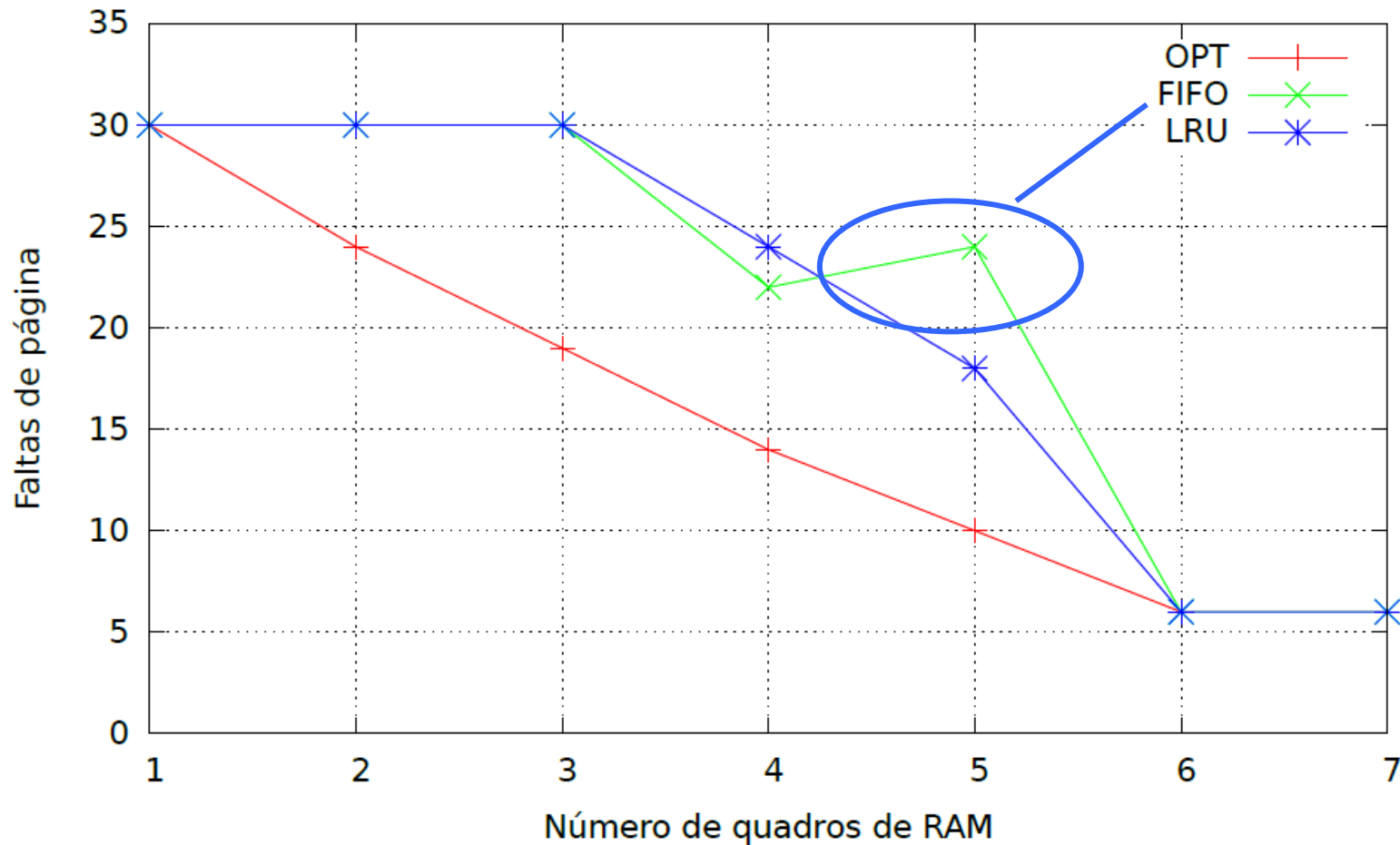
Anomalia de Belady (1969)

- Espera-se que, quanto **mais memória física** um sistema possua, **menos faltas** de página ocorram
- Esse comportamento intuitivo **não se verifica** em todos os algoritmos
- Alguns algoritmos apresentam comportamento estranho:
 - Ao **aumentar o número de quadros** de memória, o número de **faltas de página** geradas pelo algoritmo **aumenta**

Anomalia de Belady



Anomalia de Belady



An Anomaly in Space-Time Characteristics of Certain Programs Running in a Paging Machine,"
by Belady et al, *Communications of the ACM*, June 1969.

Anomalia de Belady

- Estudos demonstraram que uma família de algoritmos denominada ***algoritmos de pilha*** (à qual pertencem os algoritmos OPT e LRU, entre outros) NÃO apresenta a **anomalia de Belady**
- Ocorre basicamente no algoritmo **FIFO**

A decorative graphic on the left side of the slide. It consists of a vertical light blue bar, a vertical orange bar, and a grey rectangle. A horizontal dark blue bar extends from the grey rectangle across the top of the slide.

De volta aos algoritmos...

3. LRU – *Least Recently Used*

- LRU é pouco usado na prática:

3. LRU – *Least Recently Used*

- LRU é pouco usado na prática:
- Exigência de registrar as datas de acesso às páginas a cada leitura ou escrita na memória:
 - Difícil de implementar de forma eficiente em SW e com custo proibitivo em HW

3. LRU – Least Recently Used

- **LRU é pouco usado na prática:**
- Exigência de **registrar as datas de acesso** às páginas a cada leitura ou escrita na memória:
 - Difícil de implementar de forma eficiente em SW e com custo proibitivo em HW
- Exigência de **varrer as datas de acesso** de todas as páginas para buscar a página com acesso mais antigo:
 - Exige muito processamento

Algoritmos derivados do LRU

Na prática a maioria dos sistemas operacionais implementam **aproximações do LRU**, como as apresentadas a seguir

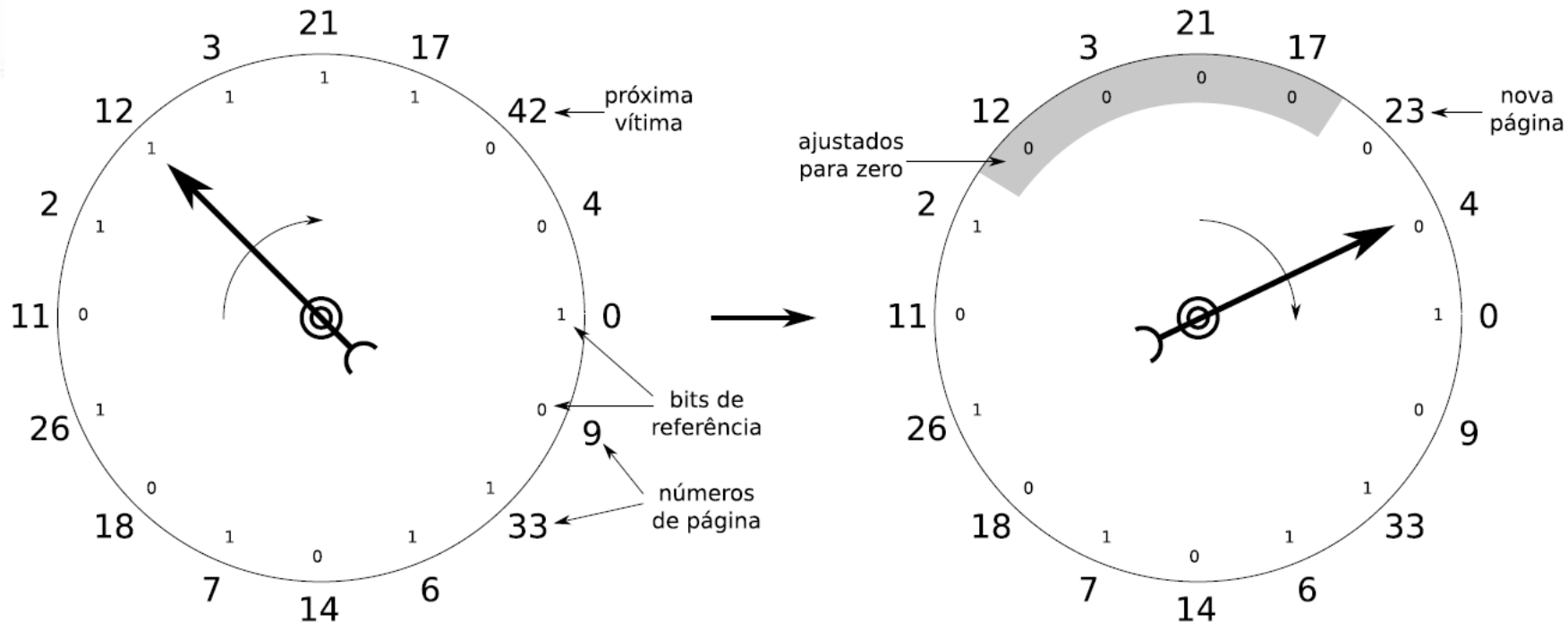
4. Segunda Chance

Uma **melhoria** simples do FIFO consiste em analisar um **bit de referência** de cada página, para saber se ela foi acessada recentemente

Em caso **afirmativo**, a página recebe uma “segunda chance”, voltando para o fim da fila com seu **bit de referência ressetado**

4. Segunda Chance

Implementação por **fila circular**:



Também conhecido como **algoritmo do relógio**

4. Segunda Chance

Caso todas as páginas sejam muito acessadas

- Todas as páginas são testadas
- Todos os bits ressetados
- A primeira página da fila será escolhida na segunda rodada de testes
→ Algoritmo FIFO

5. NRU - *Not Recently Used*

Melhora a implementação do *segunda chance*:

- Considera também o **bit de modificação**

Indica se o conteúdo de uma página foi **modificado** após ela ter sido carregada na memória

5. NRU - *Not Recently Used*

Usando os bits

- *R* (referência)
- *M* (modificação)

5. NRU - *Not Recently Used*

Usando os bits

- *R* (referência)
- *M* (modificação)

Páginas em memória são classificadas:

1. ($R = 0, M = 0$): não referenciadas e conteúdo não foi modificado
 - Melhores candidatas à substituição

5. NRU - *Not Recently Used*

2. ($R = 0, M = 1$): não foram referenciadas recentemente, mas conteúdo já foi modificado
- Não são escolhas tão boas, pois terão de ser gravadas na área de troca antes de serem substituídas

5. NRU - *Not Recently Used*

3. ($R = 1, M = 0$): referenciadas recentemente e conteúdo inalterado
- São provavelmente **páginas de código** que estão sendo usadas ativamente e **serão referenciadas novamente** em breve

5. NRU - *Not Recently Used*

4. ($R = 1, M = 1$): referenciadas recentemente e cujo conteúdo foi modificado
- São a pior escolha, porque terão de ser gravadas na área de troca e provavelmente serão necessárias em breve

5. NRU - *Not Recently Used*

O **algoritmo NRU** consiste em tentar substituir primeiro páginas do **nível 0**; caso não encontre, procura candidatas no **nível 1**...

6. Envelhecimento

Outra possibilidade de melhoria do algoritmo da segunda chance consiste em usar os bits de referência das páginas para construir contadores de acesso

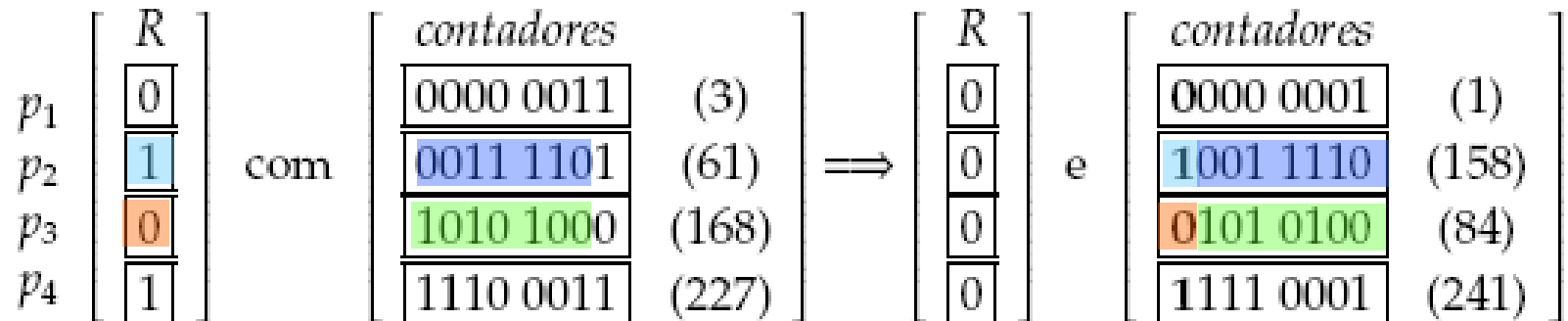
6. Envelhecimento

- A cada página é associado um **contador** inteiro com N bits
- Periodicamente, o algoritmo **varre as tabelas** de páginas
- Os **bits de referência** são lidos e **agregados** aos contadores de acessos das respectivas páginas
- Uma vez lidos, os bits de referência são **ressetados**

6. Envelhecimento

Exemplo:

Antes



Depois

- Cada contador é deslocado para a direita 1 bit, descartando o LSB
- Em seguida, o valor do bit de referência é colocado na posição do MSB

6. Envelhecimento

- A cada página é associado um **contador** inteiro com N bits
- Periodicamente, o algoritmo **varre as tabelas** de páginas
- Os **bits de referência** são lidos e **agregados** aos contadores de acessos das respectivas páginas
- Uma vez lidos, os bits de referência são **ressetados**
- A página com contador de menor valor é a escolhida