SISTEMAS OPERACIONAIS

Gerência de Memória

Análise e Desenvolvimento de Sistemas

Prof. Ms. Renato Leite

renato@sectras.edu.br

A memória principal é um componente fundamental em qualquer sistema de computação. Ela constitui o "espaço de trabalho" do sistema, no qual são mantidos os processos, threads, bibliotecas compartilhadas e canais de comunicação, além do próprio núcleo do sistema operacional, com seu código e suas estruturas de dados. O hardware de memória pode ser bastante complexo, envolvendo diversas estruturas, como caches, unidade de gerência, etc, o que exige um esforço de gerência significativo por parte do sistema operacional.

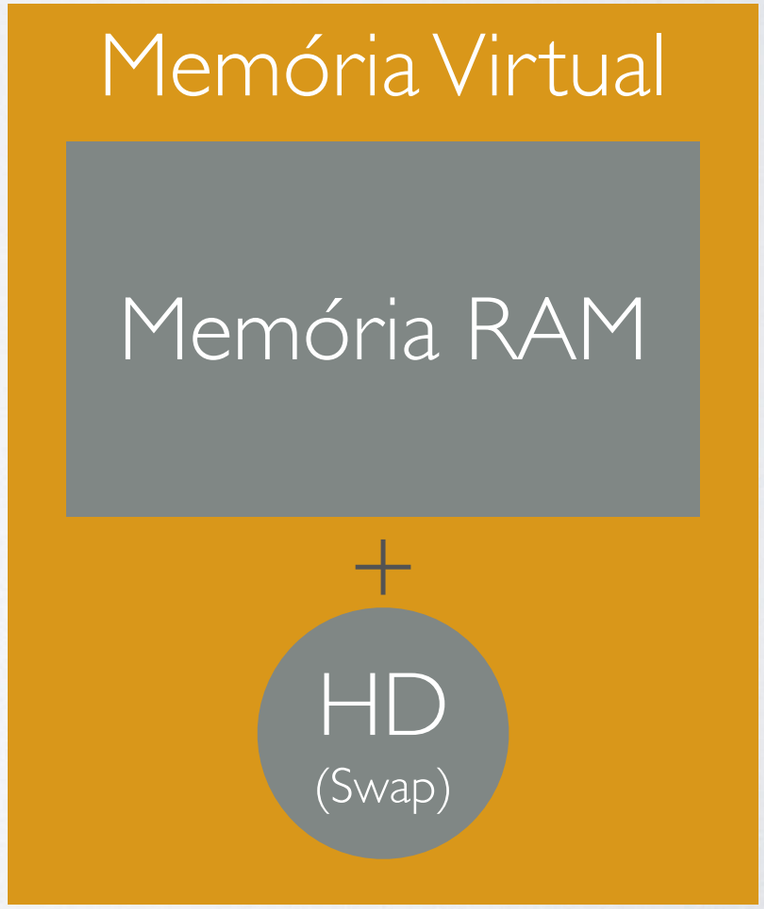
OBJETIVOS

- Conhecer o modelo de memória atualmente utilizado nos computadores;
- Compreender como o S.O. administra a alocação e divisão de processos em memória;
- As estratégias e implementações possíveis ao longo da evolução dos sistemas para particionamento de memória;
- Entender o processo de swapping.

Como expandir a memória principal cara e limitada?

- Soma de processos maior que a memória.
- Custo da memória RAM é alto;
- Processos executa e cara;
- Memórias secundárias maiores e mais baratas;

- A resposta é: Memória Virtual.



MEMÓRIA VIRTUAL

- Os processos não estão sempre sendo executados;
- Multimidia e aplicativos que manipulam grandes arquivos precisa de memória livre;
- Uma alternativa é mover dados sem uso para o HD (mais barato);
- Nada é executado fora da memória!

MEMÓRIA VIRTUAL

- Inicialmente os mecanismos de paginação envolviam a remoção do processo como um todo da memória;

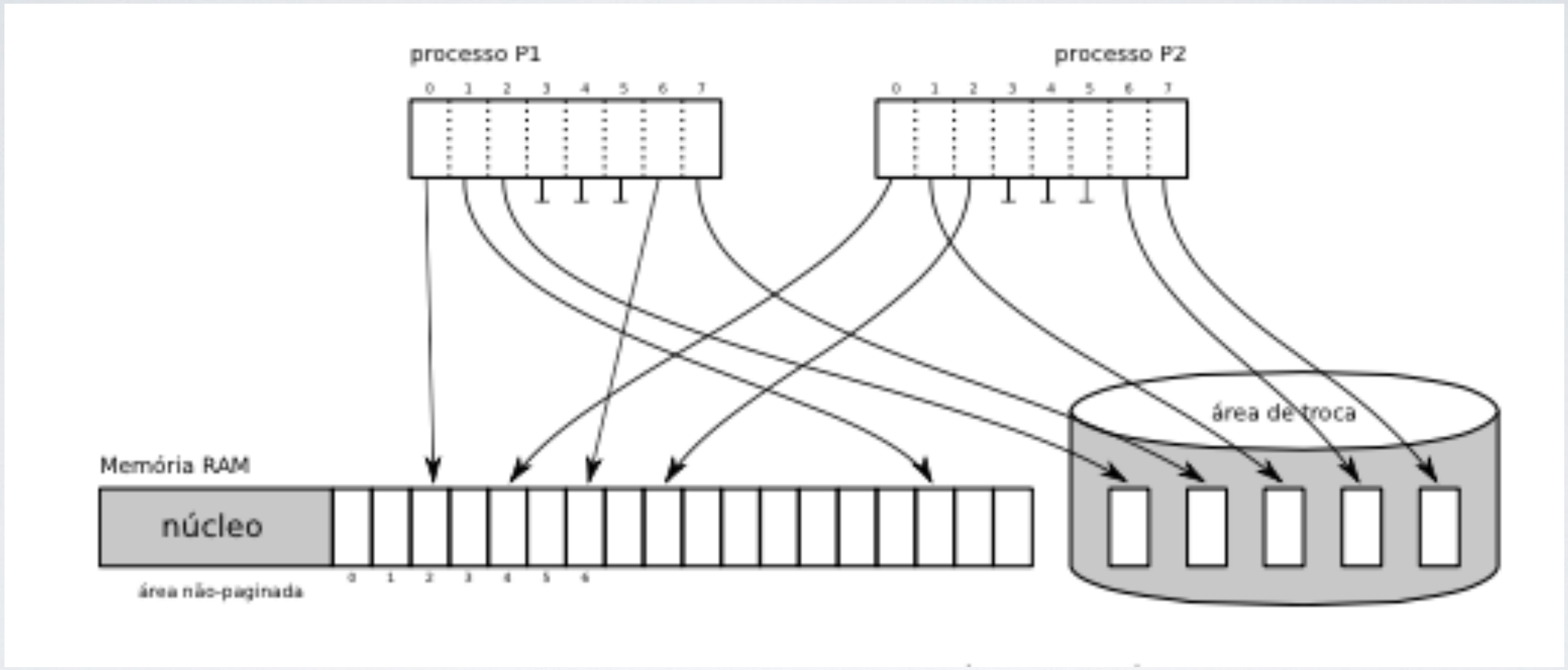
- Caso uma página não seja encontrada em memória RAM a MMU gera uma interrupção que desviará a execução para o S.O., que irá então trazer a página do HD para memória;

- Problemas?

  - Se a memória já estiver cheia?

  - Em que ponto o processo parou?

Memória Virtual

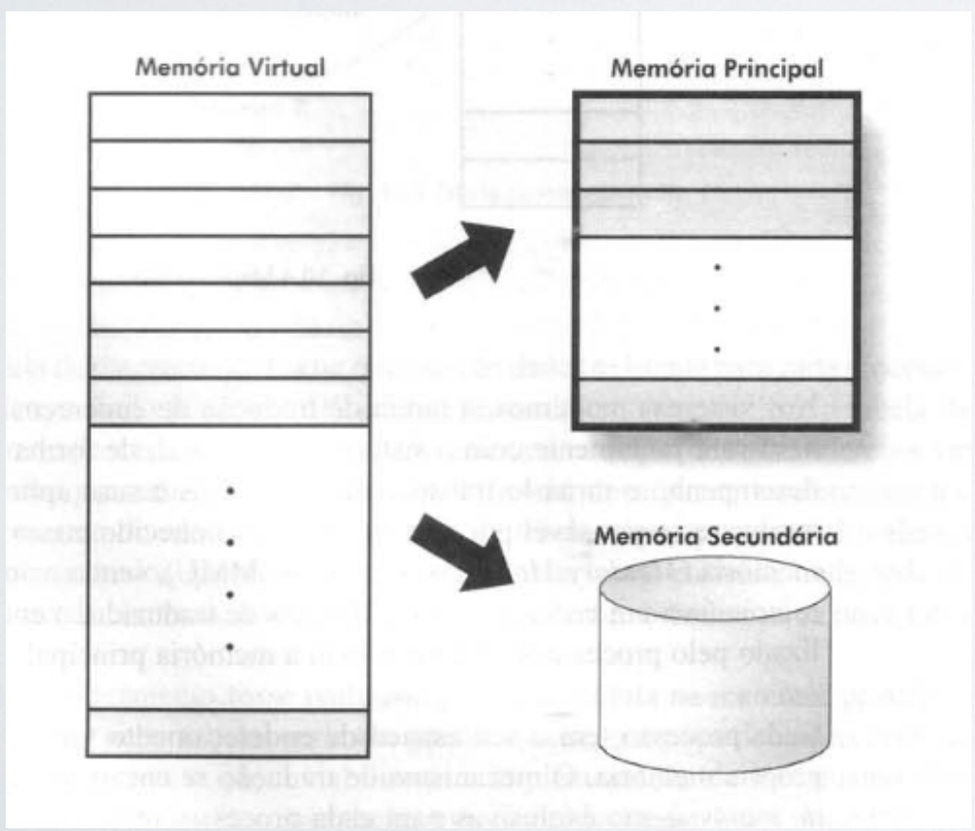


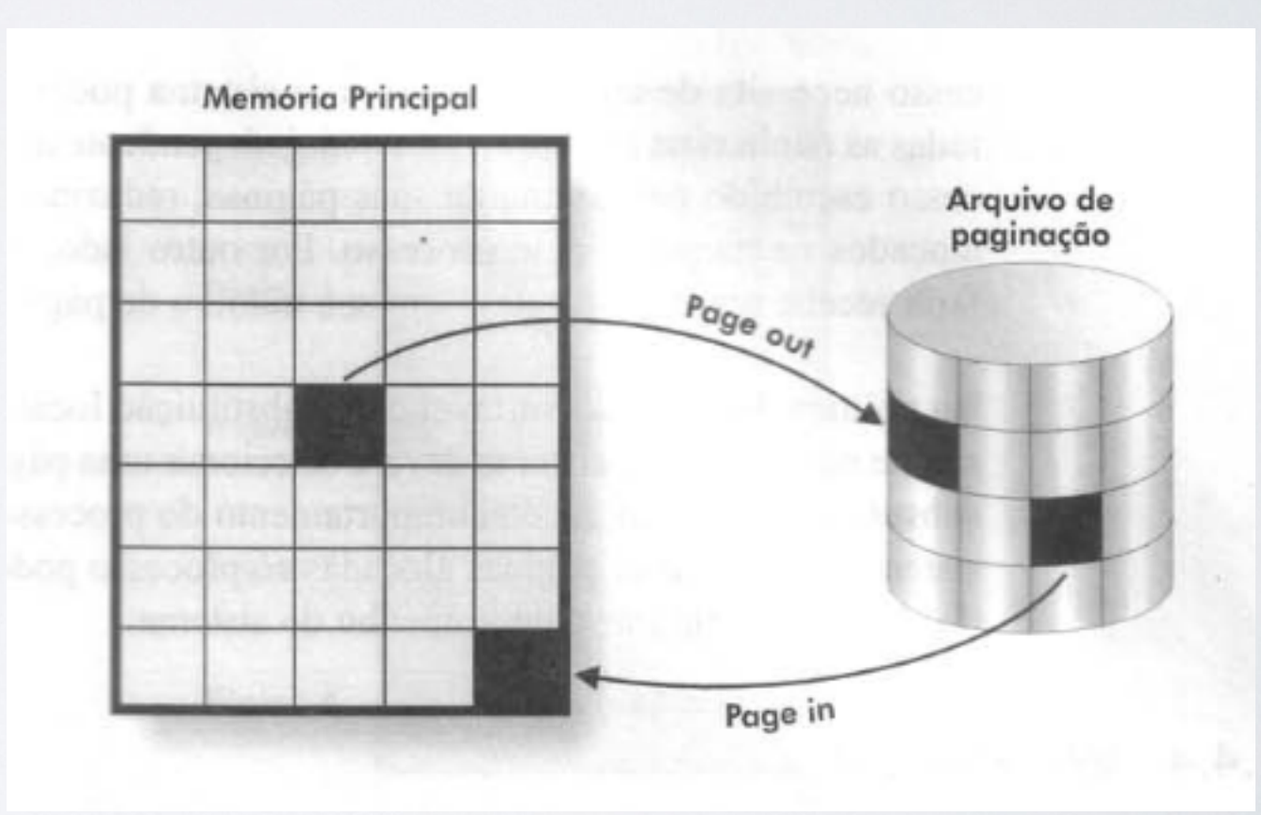
MEMÓRIA VIRTUAL

- Alguns fatores influenciam na paginação:
  - Processos que utilizam dados próximos, de forma a atingir sempre as mesmas páginas usa menos páginas e consequentemente geram menos falta de página;
  - Sistemas com pouca memória RAM levam a um ciclo de falta de páginas, gerando um ocasionando o que se chama de *trashing;*
  - Algoritmo de substituição de páginas ineficiente pode levar a remoção de páginas necessárias a execução do processo corrente, gerando swap desnecessário.

SWAP/PAGINAÇÃO

- Páginas fora da memória RAM devem ser recuperadas em disco através da técnica de SWAP;





CRITÉRIOS PARA SUBSTITUIÇÃO DE PÁGINAS

Idade da página: há quanto tempo a página está na memória; páginas muito antigas talvez sejam pouco usadas.

Frequência de acessos à página: páginas muito acessadas em um passado recente possivelmente ainda o serão em um futuro próximo.

Data do último acesso: páginas há mais tempo sem acessar possivelmente serão pouco acessadas em um futuro próximo (sobretudo se os processos respeitarem o princípio da localidade de referências).

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 ou tempo de resposta poderão ser prejudicados.

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 (dentro do arquivo executável correspondente ao processo). Por outro lado, páginas de dados que tenham sido alteradas precisam ser salvas na área de troca.

Páginas especiais: páginas contendo *buffers* de operações de entrada/saída podem ocasionar dificuldades ao núcleo caso não estejam na memória no momento em que ocorrer a transferência de dados entre o processo e o dispositivo físico. O processo também pode solicitar que certas páginas contendo informações sensíveis (como senhas ou chaves criptográficas) não sejam copiadas na área de troca, por segurança.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS

- Normalmente, quando se existe a necessidade de se efetuar um page-in, é preciso que haja page-out. Ou seja, páginas são substituídas;

- O sistema operacional pode fazer esse trabalho de paginação de diversas formas, com critérios que normalmente procuram retirar páginas da RAM que não provavelmente não venham a ser utilizadas nos próximos ciclos;

- Existem métodos de comparação entre os diversos algoritmos, em nossos estudos nos limitaremos à comparação ao algoritmo ótimo.

ALGORITMO FIFO

- First in, First out;
- Páginas mais antigas saem antes;
- Utiliza uma fila ordenada pela entrada da página na memória;
- Não utiliza atributos das páginas como referência, modificação ou prioridade.

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

ALGORITMO FIFO

Utilizando-se uma sequência de acessos a páginas vemos como o algoritmo FIFO funciona ao longo de 24 ciclos.

ALGORITMO OPT

- Algoritmo de referência, representa a melhor escolha em substituição de páginas;

- Páginas que não serão utilizadas são retiradas de memória (como saber?)

- Algoritmo não implementável;

- Todos os demais algoritmos que serão vistos se espelham em sua lógica (utilizar bit de referência para escolha das páginas que devem ser excluídas);

- No conjunto hipotético utilizado para avaliação dos algoritmos, o OPT gera 11 faltas de páginas.

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

ALGORITMO ÓTIMO

Utilizando-se uma sequência de acessos a páginas vemos como o algoritmo OPT funciona ao longo de 24 ciclos.

ALGORITMO LEAST RECENTLY USED (LRU)

- Considera como vítimas as páginas menos utilizadas pelos processos ao longo do tempo;

- Boa eficiência matemática, gerando poucas faltas de página em relação ao demais algoritmos e se aproxima do OPT;

- Verifica o momento de acesso das páginas, quais as que foram acessadas a mais tempo.

- Se diz simétrico ao OPT, pois busca as páginas que não foram usadas no passado para paginar;

- Implementável, porém não viável:

    - Necessita varrer todas as páginas em busca da mais antiga referenciada, além de armazenar todos os momentos em que as páginas foram acessadas.

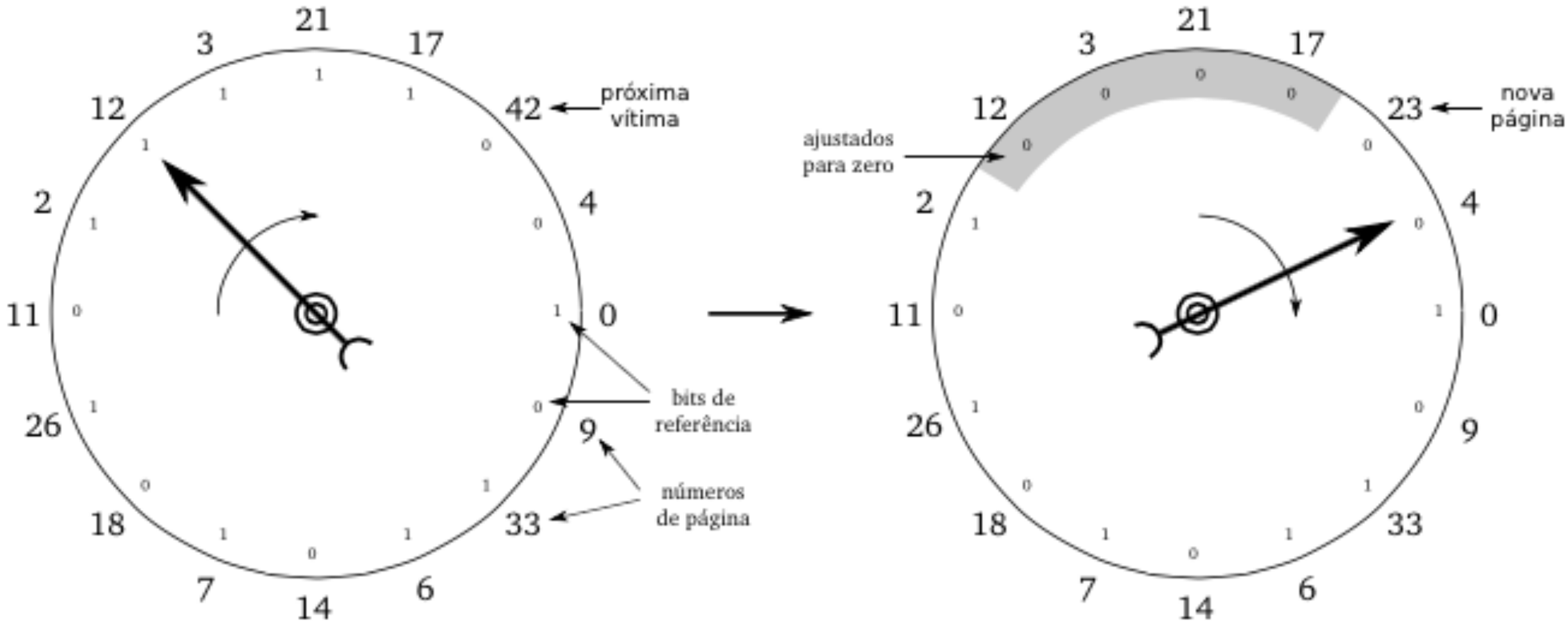
	antes				depois				
t	q_o	q_1	q_2	**página**	q_o	q_1	q_2	**faltas**	**ação realizada**
1	-	-	-	0	0	-	-	★	p_0 é carregada no quadro vazio q_0
2	0	-	-	2	0	2	-	★	p_2 é carregada em q_1
3	0	2	-	1	0	2	1	★	p_1 é carregada em q_2
4	0	2	1	3	3	2	1	★	p_3 substitui p_0 (há mais tempo sem acessos)
5	3	2	1	5	3	5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

ALGORITMO LRU

Utilizando-se uma sequência de acessos a páginas vemos como o algoritmo LRU funciona ao longo de 24 ciclos.

ALGORITMO SEGUNDA CHANCE

- Algoritmo que utiliza como critério a referência a página;
- Procura por páginas mais antigas sequencialmente (mesmo mecanismo do FIFO);
- Ao encontrar uma página acessada a pouco tempo, zera seu bit (dando-lhe uma segunda chance) para garantir que a mesma de fato não foi referenciada ao longo da próxima rodada;
- Ao encontrar uma página não referenciada na última rodada, a mesma é substituída.



ALGORITMO SEGUNDA CHANCE

ALGORITMO NOT RECENTLY USED (NRU)

- Não apenas considera o acesso, mas a modificação da página para escolher ao longo de níveis de qualidade em substituição de páginas a melhor a ser retirada da RAM;
- Percorre a memória em busca da melhor candidata à substituição;
- Utiliza também uma lista circular;
- Considerando os 2 bits utilizados no cálculo temos:

Qualidade	Referência	Modificação
Ótima	0	0
Boa	0	1
Ruim	1	0
Péssima	1	1

ALGORITMO ENVELHECIMENTO

- Utiliza bit de referência;

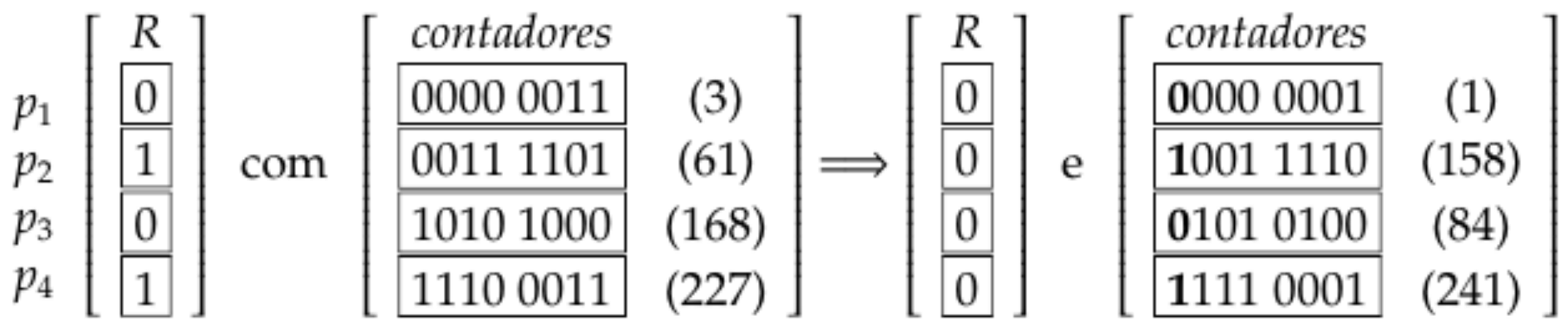
- A cada rodada do algoritmo ao longo das páginas, a referência ao quadro é utilizada para manter a mesma “jovem";

- Utiliza um conjunto de bits (normalmente 8 ou 1 Byte) para cálculo da idade da página:

  - Não se pode simplesmente somar mais a cada vez que se acessa a página (overflow);

  - Utiliza-se um esquema de deslocamento de bits a direta a cada rodada (tornando todas páginas mais velhas);

  - Caso a página tenha sido referenciada, a mesma é renovada pelo MSB setado para 1.

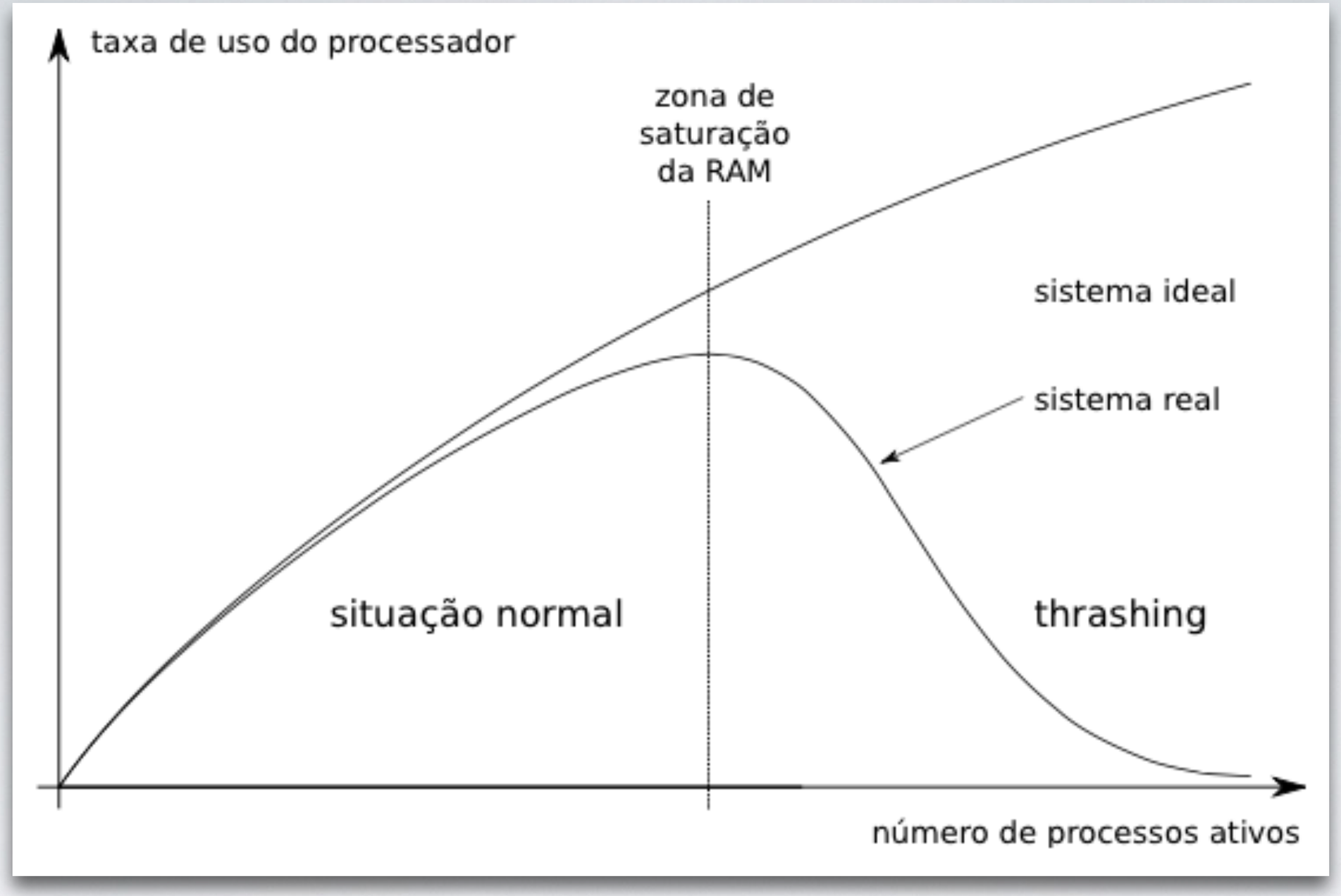


ALGORITMO ENVELHECIMENTO

Considerando 4 páginas antes e depois da atuação do algoritmo de envelhecimento. Notem como as páginas que foram referenciadas são renovadas e vejam como as não referenciadas perdem vigor e envelhecem.

WORKSET E TRASHING

- Todo processo possui um conjunto de trabalho, que nada mais é que as páginas que estão em RAM - Quadros, que ele utiliza para executar suas tarefas;
    - Caso o processo respeite o princípio da localidade de referências, esse workset irá variar de forma lenta e existirá pouca paginação;
- Ao passo que a memória RAM chega ao seu limite, existirão mais paginações e caso essa situação perdure, pode vir a ocorrer o fenômeno já citado como trashing:
    - A quantidade de processos ativos mantendo seus conjuntos de trabalho em RAM não é suportado pela limitação de memória, gerando a cada troca de contextos uma interrupção para carregamento de páginas que estão em disco.
    - Alternativas: aumentar tempo de processador, pausar/matar processos ou pra prevenir: aumentar a RAM, limitar a quantidade de processos máximo.



TRASHING

Vejam como a memória RAM em virtude de sua limitação obrigado a troca de páginas constantes, mantendo diversos processos em pausa e praticamente inutilizando a máquina.

REFERÊNCIAS

- MACHADO, Francis B.; MAIA, Luiz Paulo. Arquitetura de Sistemas Operacionais. 3a ed. Rio de Janeiro : LTC, 2002.

DÚVIDAS?