SISTEMAS OPERACIONAIS

ADS

SANDRO ROBERTO ARMELIN

Fatec Mogi Mirim Arthur de Azevedo

GERENCIAMENTO DE MEMÓRIA

MEMÓRIA PRINCIPAL (RAM)

- Uma das maiores preocupações no desenvolvimento de S.O.
- Recurso **importante**, **escasso** e de **alto custo**.
- **Importante** porque o processador executa somente instruções localizadas na memória principal.
- Todo programador deseja memória: **Grande**, **rápida** e **não volátil**.

MEMÓRIA PRINCIPAL (RAM)

- Em S.O. mono programáveis a gerencia de memória não é muito complexa.
- Já em S.O. multi programáveis essa gerencia se torna crítica devido necessidade de se maximizar **quantidade de usuários** e **aplicações** utilizando de forma eficiente o espaço da memória principal.
- Quais objetivos da gerencia de memória?

GERÊNCIA DE MEMÓRIA

- Programas são armazenados em **memória secundária (disco)**. Meio não volátil, de baixo custo e com recurso abundante.
- Como processador executa o que esta em memória principal (RAM), o S.O. deve transferir programas para a RAM antes de executar.
- Como o tempo de acesso a Disco é muito superior a RAM, o S.O. deve buscar reduzir o número e operações E/S ao disco.
- **Gerencia de memória**: **Deve tentar manter na memória principal maior número de processos residentes; Gerenciar de modo eficiente; manter o controle de quais partes estão em uso e quais não estão, alocando memória aos processos quando eles precisam e liberando quando eles terminam**.

GERÊNCIA DE MEMÓRIA

- Controla o uso da memória;
- Cuida de quais partes da memória estão em uso e quais estão livres;
- Aloca memória aos processos quando eles precisam;
- Desloca memória quando os processos não precisam mais;
- Gerencia a troca dos processos entre memória principal e disco.
- Mesmo na ausência de espaço livre, deve permitir que novos processos sejam aceitos e executados. Como isso é possível? → continua...

GERÊNCIA DE MEMÓRIA

- É possível através da transferência temporária de processos residentes ne memória principal para a memória secundária, liberando espaço para novos processos. **Técnica de swapping**.
- Outro ponto é permitir execução de programas maiores do que a memória física disponível. **Técnica de memória virtual**.

A MEMÓRIA É SUFICIENTE?

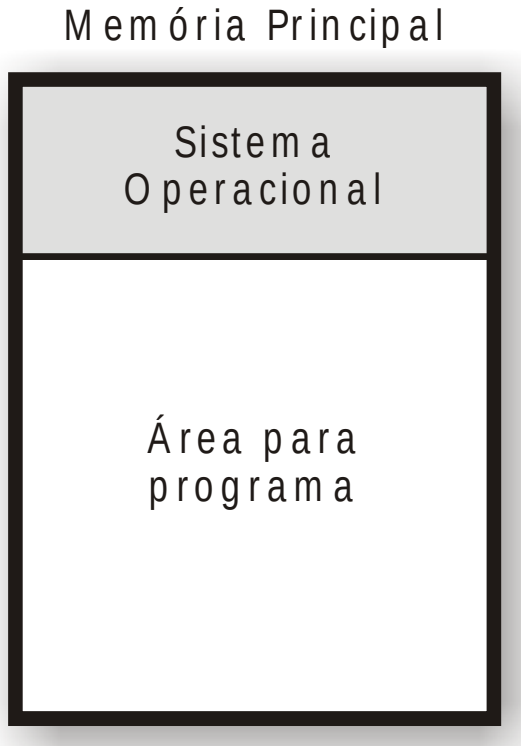
- Se a memória física do computador for **grande o suficiente para armazenar** todos os **processos ou programas** em execução, não precisaríamos falar de gerencia de memória.
- Na prática, a **quantidade de RAM** para **todos os processos** é frequentemente muito menor do que ela pode comportar.
- Dois métodos para lidar com sobrecarga de memória é utilizado:
- *Swapping*: Troca de processos abertos entre memória e disco. Processos ociosos são armazenados no disco.
- *Memoria virtual*: permite que programas possam ser executados mesmo que estejam apenas parcialmente carregados na RAM.

ESQUEMAS DE GERENCIAMENTO DE MEMÓRIA.

- Estudaremos alguns esquemas diferentes de gerenciamento de memória, desde os mais simples ao mais sofisticado.
- Objetivo é compreender os esquemas de gestão de memória.

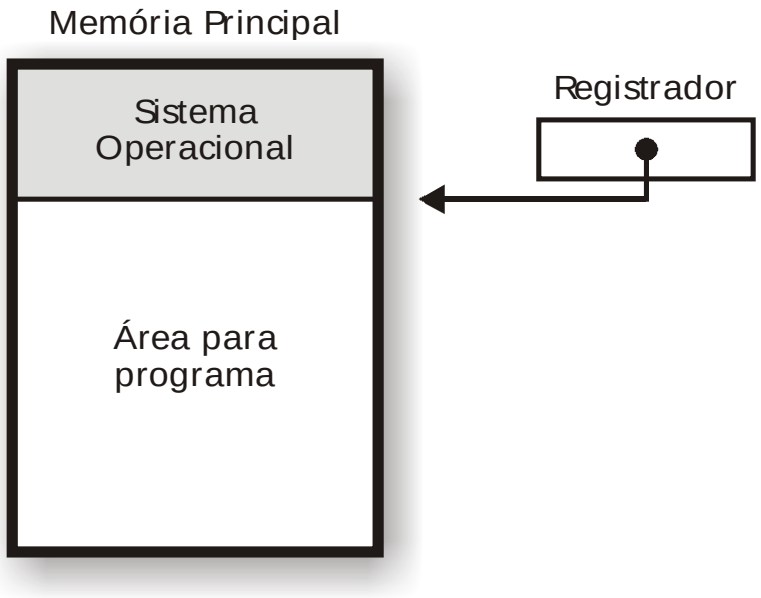
ALOCAÇÃO CONTÍGUA SIMPLES

- Implementada nos primeiros S.O.
- Divide memória em duas áreas: uma para S.O. e outra para os programas.
- Deve-se desenvolver programas com preocupação de não ultrapassar o espaço de memória disponível.
- Não permite utilização eficiente dos recursos computacionais.



ALOCAÇÃO CONTÍGUA SIMPLES

- Usuário pode ter acesso a qualquer área da memória até mesmo a área do S.O.
- Para isso não ocorrer implementa-se proteção através de um registrador, que delimita as área do SO e do Usuário.
- Ao alocar recurso o SO verifica se não esta dentro do limite do registrador.
- Caso programa não preencha totalmente a memória, existirá um espaço sem utilização.
- Os programas estão limitados ao tamanho da memória principal disponível.
- Solução para este método foi dividir programas em partes (módulos), de forma que pudesse executar módulos independentes (Overley).

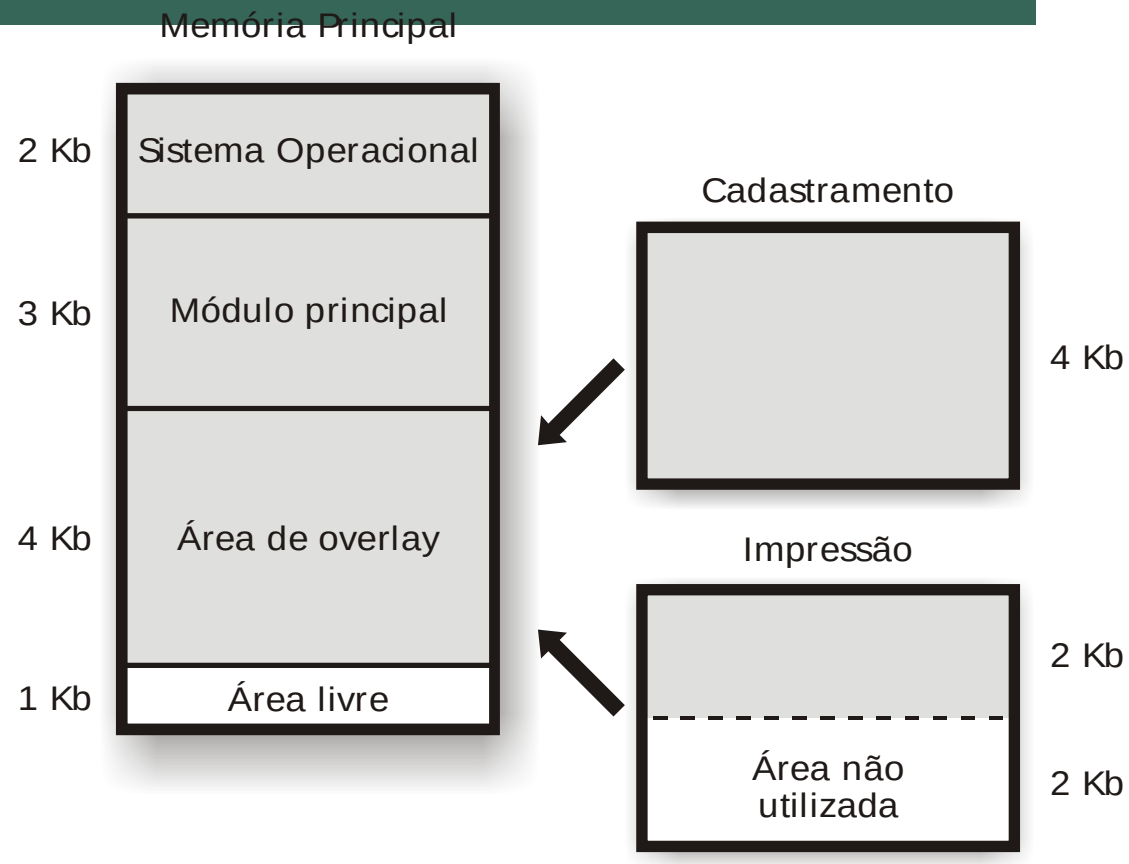


TÉCNICA DE OVERLEY

- Divisão do programa em módulos, de forma que seja possível execução independente de cada módulo, utilizando mesma área de memória.
- Exemplo: Considerar programa com 3 módulos independentes. Módulo principal comum para os outros dois módulos: Módulo principal, Módulo cadastro e módulo impressão.
- A definição das áreas de Overley é função do programador, através de comandos específicos da linguagem.
- **Técnica permite o programador expandir os limites da memória.**

TÉCNICA DE OVERLEY.

- 9Kb seria insuficiente para a memória.
- Modulo principal ocupa 3kb e a área de overlay é criada para os módulos que serão comuns no maior tamanho.
- Será utilizado a área de overlay pelos dois outros módulos quando necessário pelo modulo principal.
- Outro processo nunca acessa a área de overley do processo concorrente.
- 2Kb de área não utilizada...



ALOCAÇÃO PARTICIONADA.

- Na evolução dos S.O. e com necessidade de melhor aproveitamento dos recursos disponíveis surgiram novos métodos de alocação de memória.
- Para a multiprogramação ser eficiente foi necessário vários programas esteja na memória ao mesmo tempo, evoluindo desta forma a organização de memória principal.
  - **Alocação particionada Estática**
  - **Alocação particionada dinâmica.**

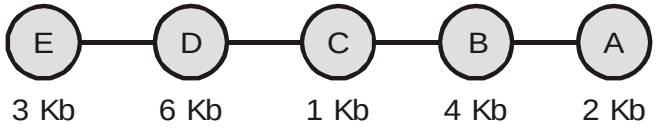
ALOCAÇÃO PARTICIONADA ESTÁTICA

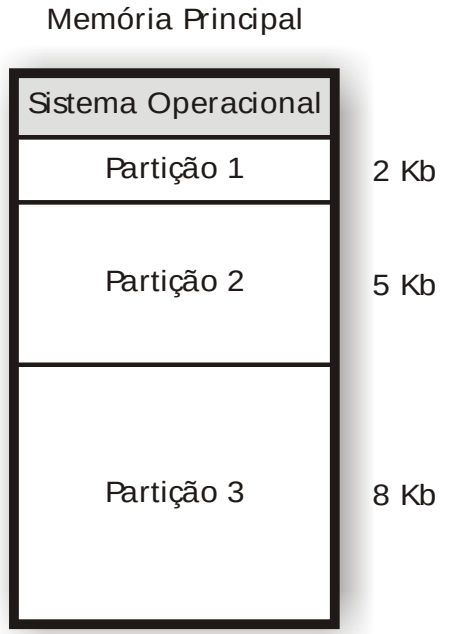
- Memória dividida em **pedaços** de tamanho fixo, chamada partições.
- Tamanho das partições estabelecido na fase da inicialização do sistema.
- Levava em consideração os programas executados anteriormente no ambiente.
- Programas deveriam ser carregados em partições específicas, dentro dos limites do seu tamanho.
- **Desvantagem: Problema fragmentação interna.**

Tabela de partições

Partição	Tamanho
1	2 Kb
2	5 Kb
3	8 Kb

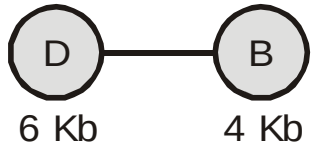
Programas a serem executados:

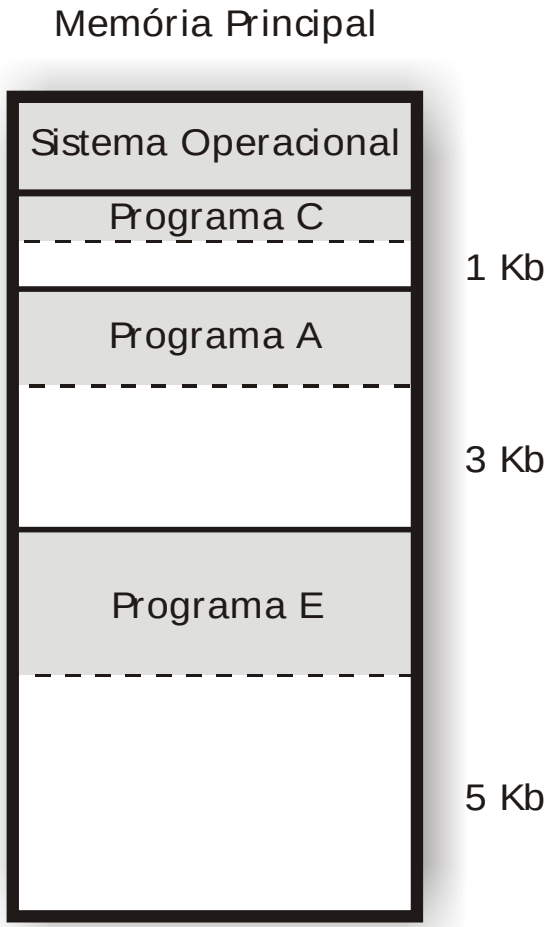




FRAGMENTAÇÃO INTERNA.

- Programas normalmente não ocupam totalmente as partições onde são carregados.
- **Este problema, decorrente da alocação fixa das partições é conhecido como fragmentação interna.**



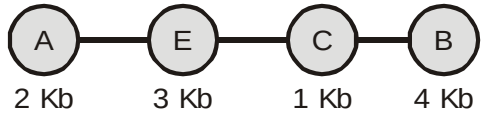


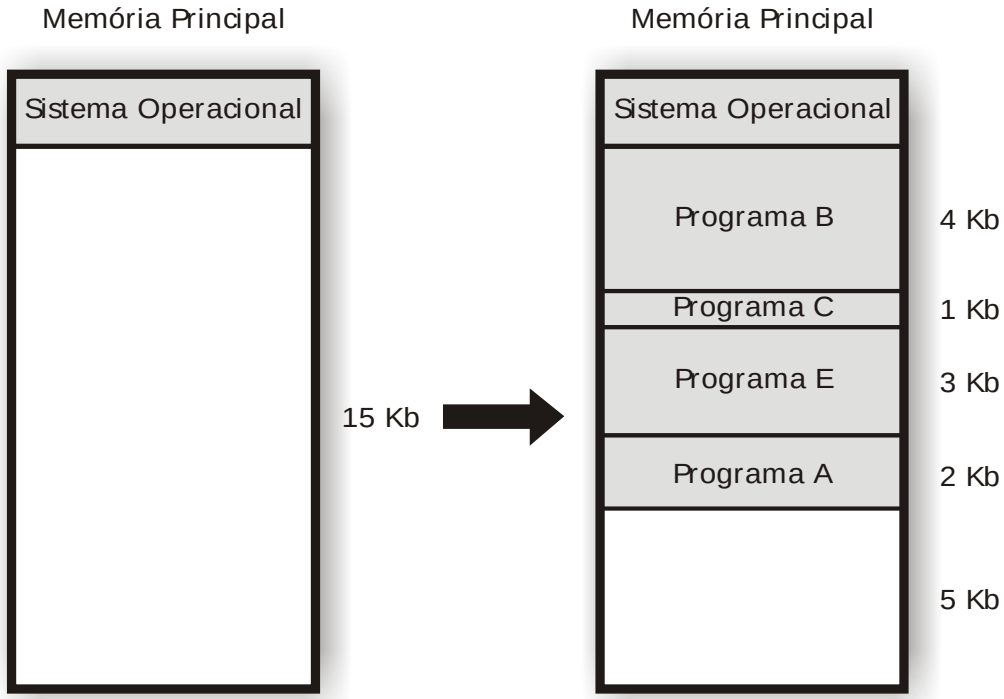
ALOCAÇÃO PARTICIONADA DINÂMICA

- Alocação estática deixou evidente necessidade de uma nova forma de gerencia de memória para minimizar a fragmentação interna.
- Eliminado o conceito de partições fixas.
- Cada programa utiliza o espaço necessário

desde que existe este espaço.

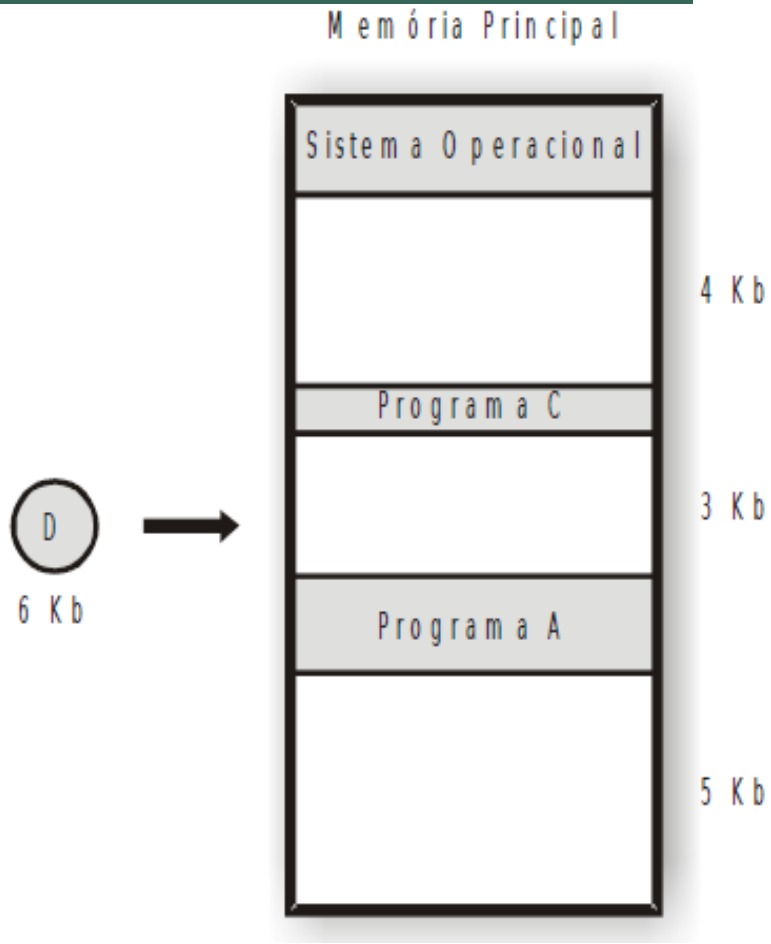
- Não ocorre problema fragmentação interna.





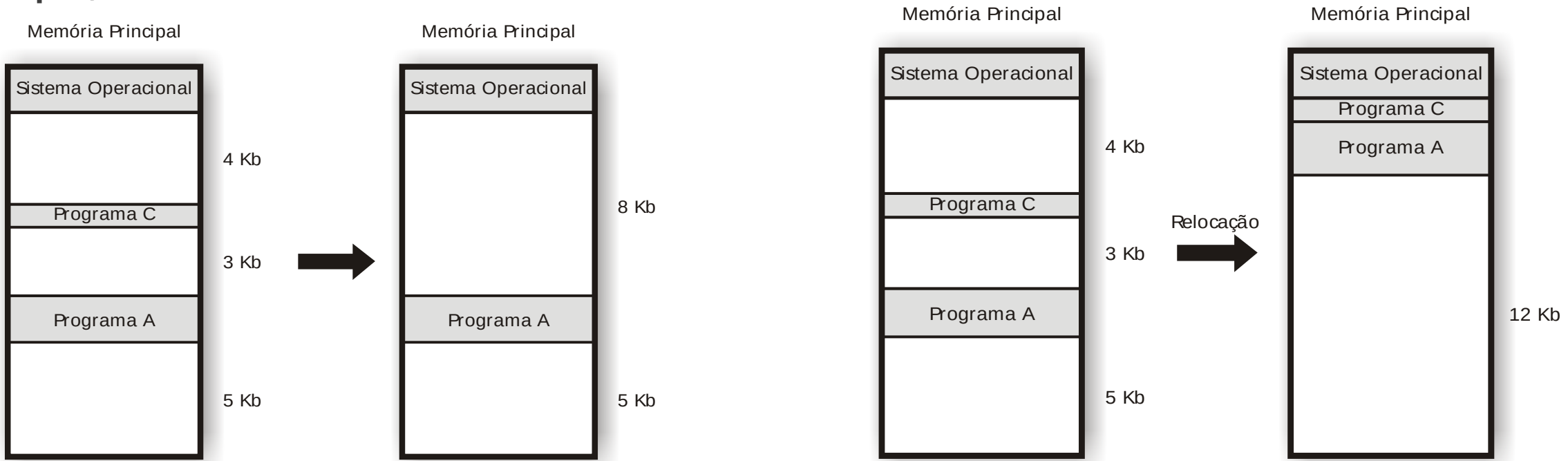
ALOCAÇÃO PARTICIONADA DINÂMICA

- Neste esquema também ocorre fragmentação.
- Ela aparecerá na medida que os processos forem terminando e deixando espaços cada vez menores na memória, não permitindo o carregamento de outros processos
- **Novo problema**: **FRAGMENTAÇÂO EXTERNA** – quando o programa deixa a memória e fica um fragmento dele quando a utilizou.



SOLUÇÕES PARA PROBLEMAS DE FRAGMENTAÇÃO

- **Solução 1** – Conforme os programas terminam, os espaços livres adjacentes são reunidos. No exemplo programa C finalizou...
- **Solução 2** – Realocação de todas as partições ocupadas, eliminando os espaços entre elas. Criando uma única área livre.

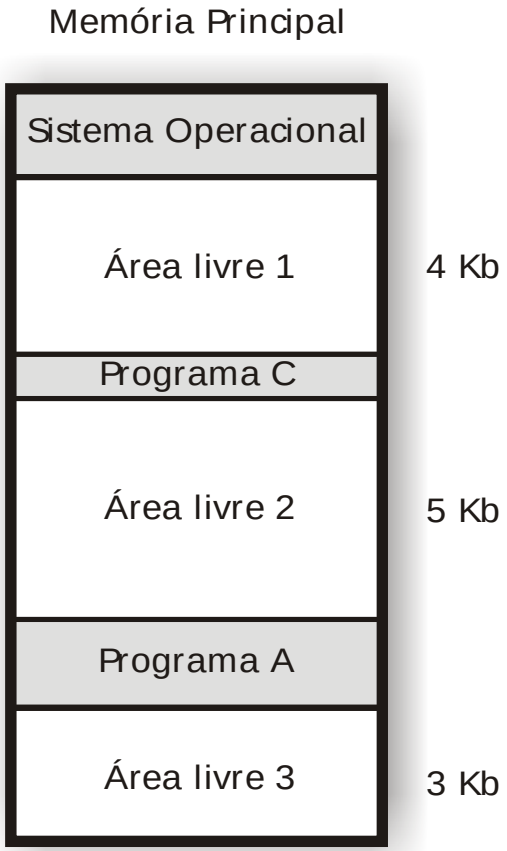


ESTRATÉGIA DE ALOCAÇÃO DE PARTIÇÃO.

- **Estratégias que os SO implementam** para determinar em qual área livre um programa será carregado para execução.
- Tenta evitar **problema de fragmentação externa**.
- A melhor estratégia a ser adotada pelo SO depende de vários fatores. O mais importante é tamanho dos programas.
- Sistemas possui lista de áreas livres, com tamanho de cada área.

LISTA DE ÁREAS LIVRES.

Áreas livres	Tamanho
1	4 Kb
2	5 Kb
3	3 Kb

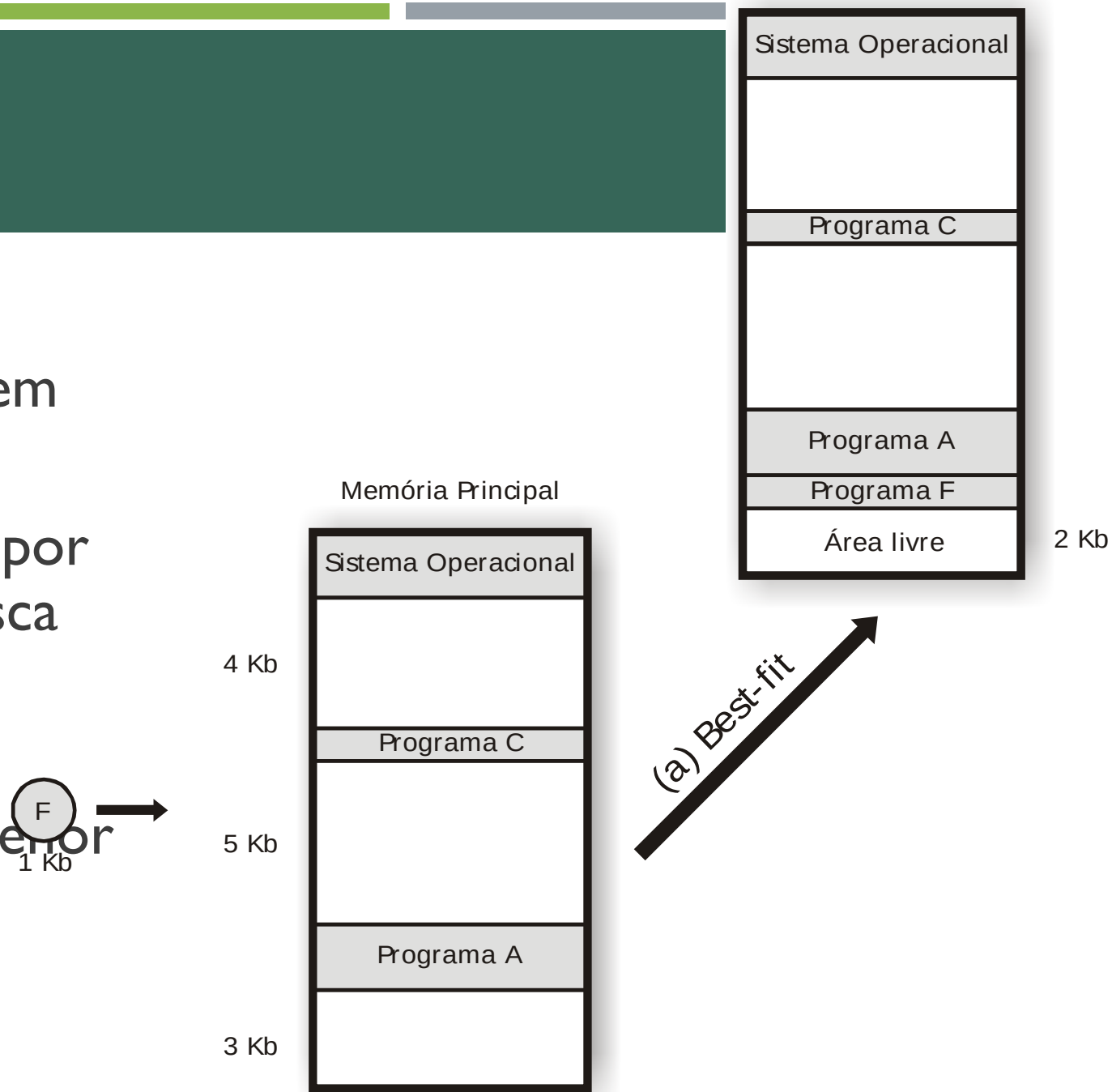


ESTRATÉGIAS DE ALOCAÇÃO.

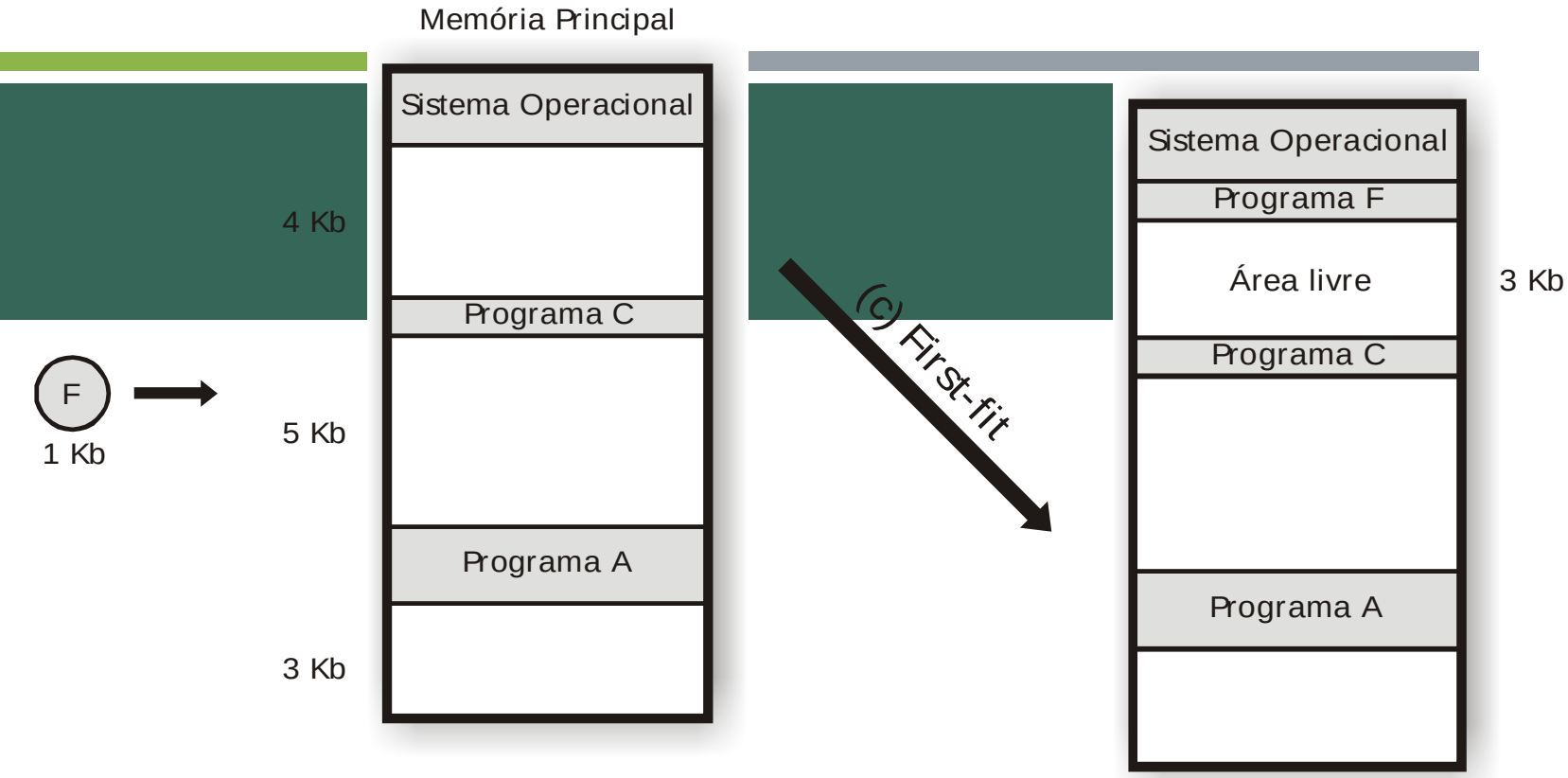
- Os “buracos” são escolhidos pelos processos através de algoritmos:
- **First Fit** (primeiro encaixe): percorrer a fila até encontrar o primeiro espaço em que caiba o processo.
- **Best Fit** (melhor encaixe): consiste em verificar toda a lista e procurar o espaço que tiver mais próximo das necessidades do programa, que deixa e menor espaço sem utilização.
- **Worst Fit** (pior encaixe): pega sempre o maior espaço disponível.

BEST-FIT
MELHOR ENCAIXE

- Partição é escolhida em que o programa deixa o menor espaço sem utilização.
- Lista de áreas livres esta ordenada por tamanho, diminuindo tempo de busca por área livre.
- Desvantagem: aumenta problema fragmentação pois como aloca a menor partição contribui para deixar pequenos espaço sem utilização.

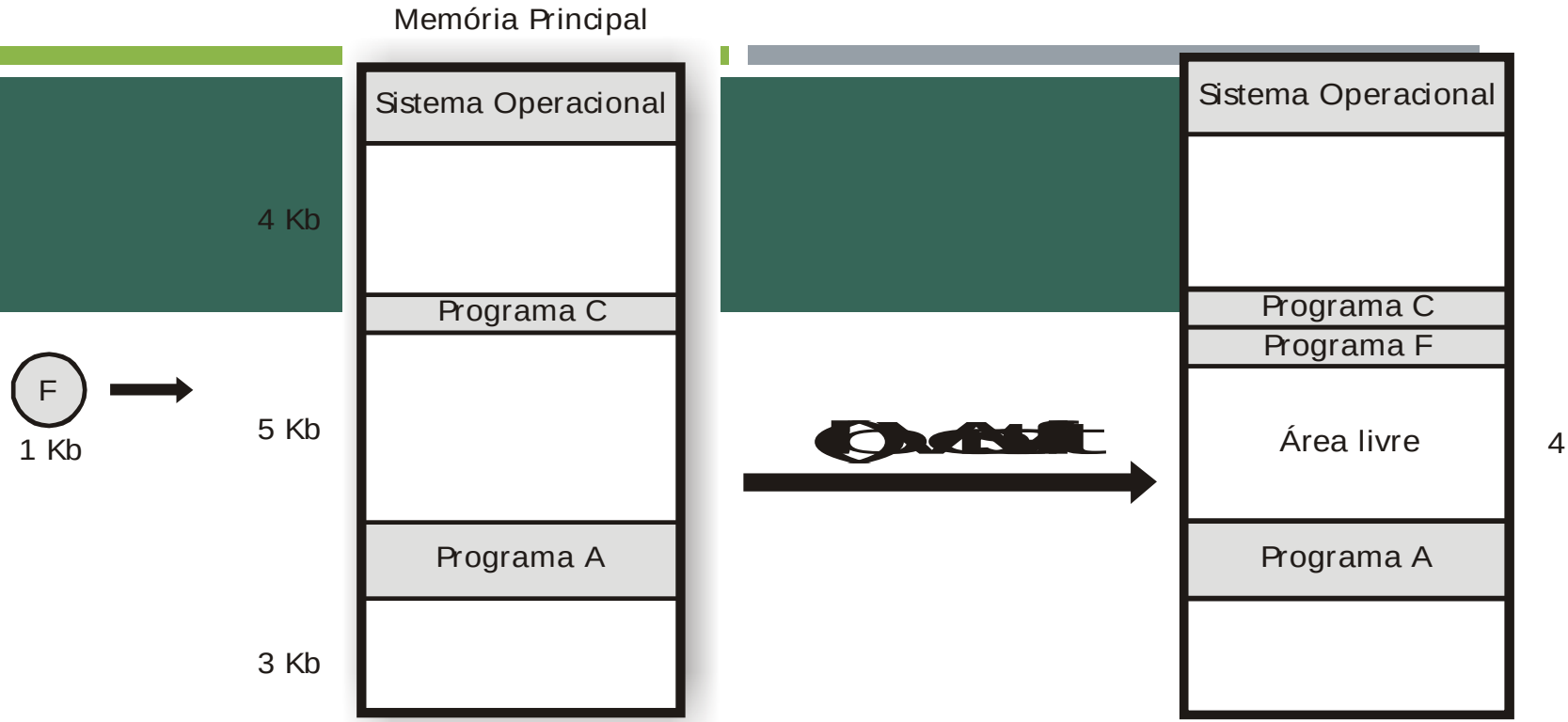


FIRST-FIT
PRIMEIRO ENCAIXE



- Primeira partição com tamanho suficiente para carregar o programa é escolhido.
- A lista de área livre esta ordenada por endereço.
- Estratégia mais rápida.

WORST-FIT
PIOR ENCAIXE



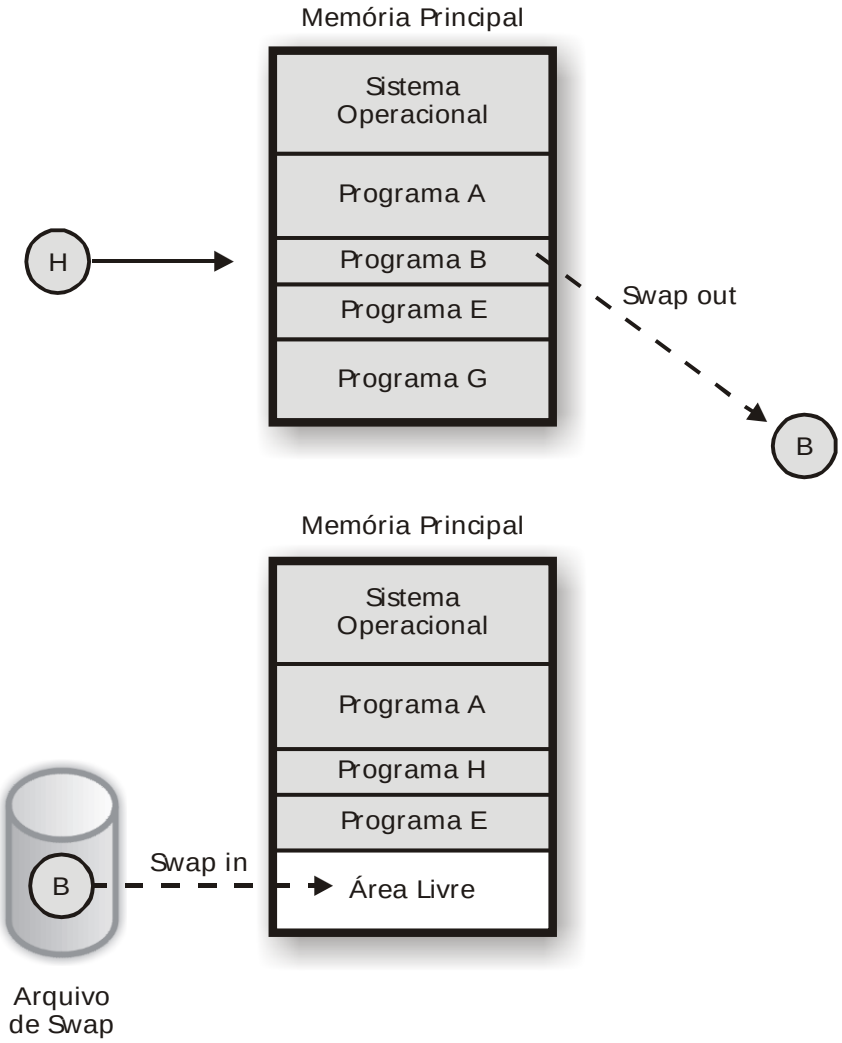
- Pior partição representa a que deixa o maior espaço sem utilização.
- Apesar de utilizar maiores partição, a técnica deixa espaços livres maiores para permitir outros programas utilizar memória. Diminuindo assim problemas de fragmentação.

SWAPPING

- **Técnica com objetivo de resolver o problema de insuficiente de memória** para execução todos os programas.
- Nos esquemas apresentados até agora o processo fica na memória durante toda sua execução, inclusive em caso de I/O, por exemplo.
- **Swapping** – processos esperam por memória livre para serem executados. Combinação de memória principal e secundária.

SWAPPING.

- Gerencia de memória escolhe processo residente da memoria principal para secundária (swap out).
- Posteriormente processo carregado de volta (swap in), onde pode continuar a execução.
- Algoritmo de escolha do processo a ser retirado deve priorizar com menor chance de execução, para minimizar swapping desnecessário.



MEMÓRIA VIRTUAL.

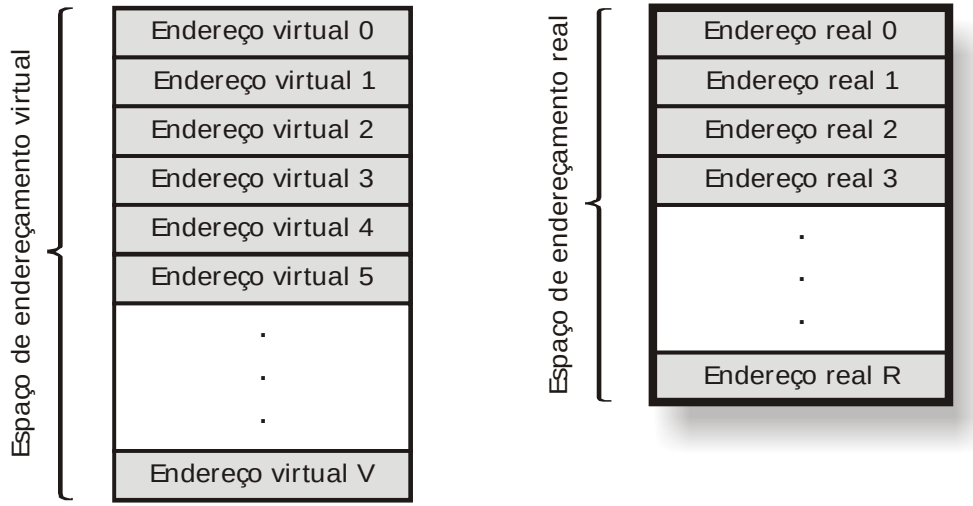
- As implementações **vistas anteriormente** no gerenciamento de memória se mostraram muitas vezes ineficientes.
- **Memoria Virtual** – Técnica sofisticada de gerencia de memória onde memória principal e secundária são combinadas.
- **Ilusão de memória de maior capacidade do que a real**.
- Forte relacionamento entre a gerencia de memória virtual e a arquitetura de ***hardware*** dos ***Sistemas Operacionais***. Por motivo de desempenho, comum algumas função da memória virtual ser implementado diretamente no hardware.

MEMÓRIA VIRTUAL.

- **Compartilhamento da RAM de forma eficiente.**
- **Técnica que utiliza o conceito de Swapping mais a combinação entre memória principal (Ram) e secundária (Disco).**
- **Estratégia de Swapping** – consiste em trazer cada processo para a memoria e executa-lo.
- **Estratégia memoria virtual** – permite que programas possam ser executados mesmo que estejam parcialmente carregados na memoria principal.

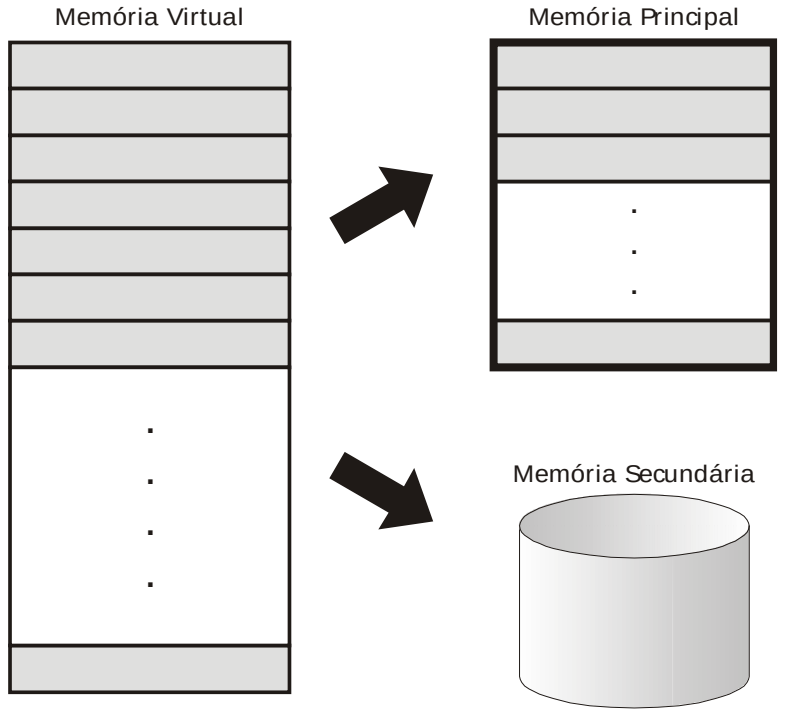
ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- Conceito Memória virtual utiliza abstração semelhante a um vetor (linguagem alto nível).
- Fazer referência a um elemento do vetor não há preocupação em saber a posição de memória do dado. O compilador se encarrega de implementar o recurso – transparente ao programador.
- Em Memória Virtual um programa faz referência a endereços virtuais



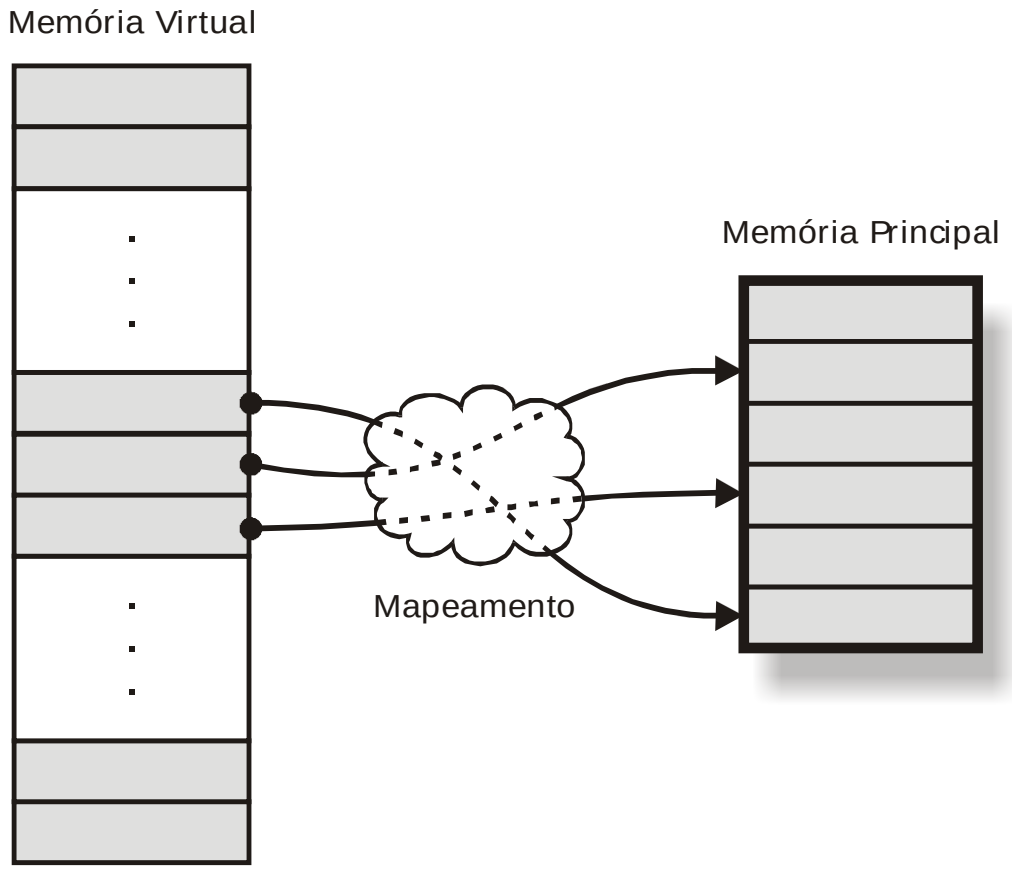
ESPAÇO DE ENDEREÇAMENTO VIRTUAL

- S.O. utiliza memória secundária como extensão para memória principal.
- Programas são divididos em partes pelo S.O.
- Somente uma parte do programa fica em residente na principal.
- Um programa pode fazer referencia a endereços virtuais que esteja fora dos limites da RAM – Como SO sabe onde esta parte do processo?



MAPEAMENTO

- Processador apenas executa instrução e referencia dados no espaço de endereçamento real (RAM).
- Mapeamento transforma endereço virtuais em reais.
- Sistemas Operacionais modernos a tradução de endereços realizado pelo hardware junto ao S.O. através dispositivo hardware MMU (Memory Management Unit)

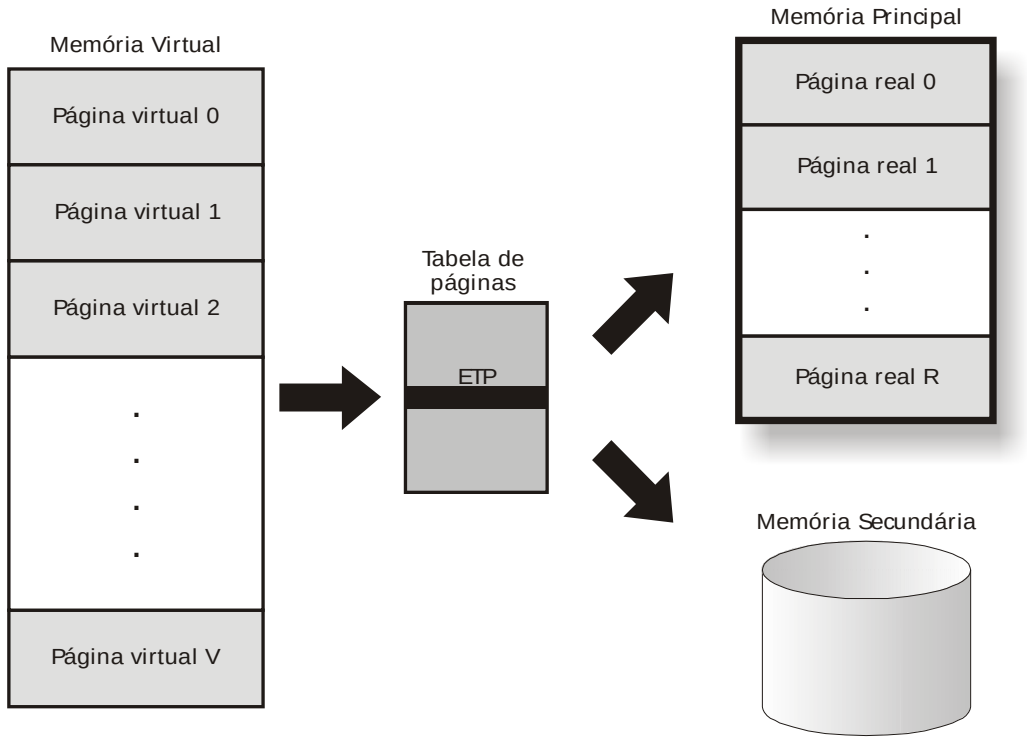


MEMÓRIA VIRTUAL POR PAGINAÇÃO

- Técnica onde os espaços de endereçamento virtual e real são divididos em blocos de mesmo tamanho chamado de páginas.
- Páginas virtuais para o espaço virtual e Frames ou páginas reais para o espaço real (RAM).

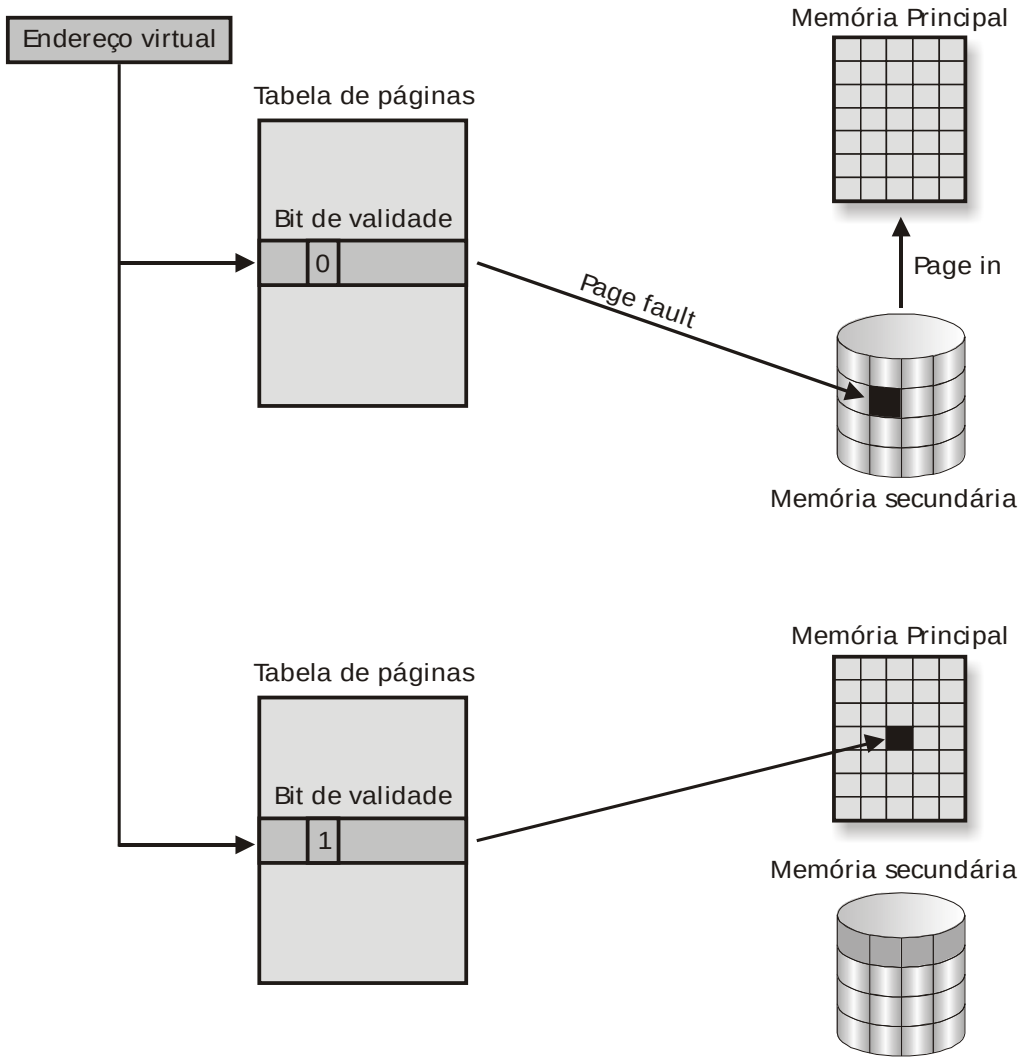
MEMÓRIA VIRTUAL POR PAGINAÇÃO

- Quando um programa é executado, as páginas virtuais são transferidas da memoria secundária para a principal e colocadas nos frames.
- Sempre que um programa fizer referencia a endereço virtual, mecanismo localizara na ETP (Entrada na tabela de páginas) o endereço do frame que se encontra no endereço real correspondente.
- E ele pode estar na principal ou secundária.



MECANISMO DE TRADUÇÃO

- BIT de validade – indica se a pagina esta na memória (1), ou na secundária (0).
- Quando fazer referencia a um processo que não esta na RAM ocorre PAGE FAULT.
- Quando Bit validade for 0, S.O. transfere a página da secundária para principal. É um processo de E/S conhecido como Page in ou paginação.
- O Nº de page faults gerado pela gestão de memória define sua taxa de paginação.



PAGE FAULT – FALTA DE PÁGINA.

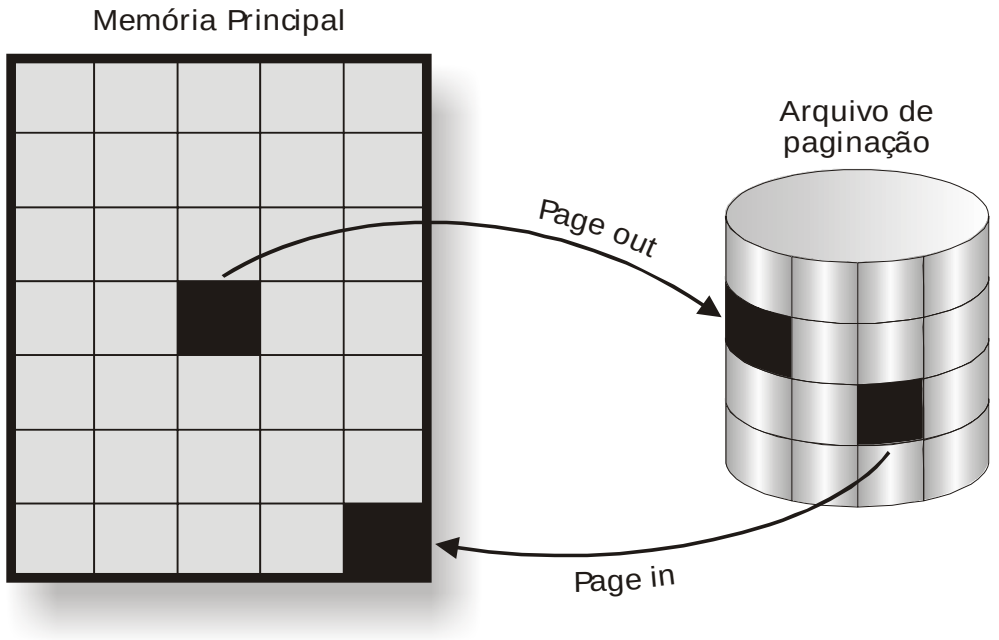
- **Page fault** – Quando a pagina não esta na memoria principal.
- Quando ocorre uma falta de página, o SO precisa escolher uma página a ser removida da memória a fim de liberar espaço para uma nova página. Neste caso entra em ação o algoritmo de substituição de página.

TAXA PAGINAÇÃO.

- Taxa de paginação: N° de Page faults gerado pelos processos em determinado tempo.
- Overhead (custo) gerado pelo mecanismo paginação é **inerente da gerência de memória**, porem taxa de paginação elevada pode **comprometer o escalonamento** de processos com alta taxa de E/S.
- Quando ocorre page fault o processo que precisou da pagina passa de execução para espera, aguardando a pagina ficar disponível na RAM.

POLÍTICA DE SUBSTITUIÇÃO DE PÁGINAS.

- Possui objetivo dentre as diversas páginas alocadas pelos processos qual deverá se liberada.
- Decisão ocorre através dos algoritmos de substituição de página.



WORKING SET

- Devido divisão programas em diversas páginas, para evitar referencias a páginas que não esteja na memória.
- **Working Set –** S.O. conhecer quais páginas comum aos programas e através principio localidade, processador tende a concentrar suas referencias a um **conjunto** de **página** durante período de tempo.
- Inicio execução de um programa ocorre elevado número de page faults. Com o tempo isso diminui.
- Tem objetivo de minimizar problema de thrashing.

TRASHING

- Ocorrência de um número elevado de page faults.
- Consequentemente ocorre inúmeras operações de E/S.
- Problemas para o desempenho do sistema.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA.

- **Maior problema** gerência de memória virtual por paginação não é decidir quais páginas carregar para RAM e sim ***quais liberar***.
- Quando processo precisa de página na memória principal e não possuem **frames disponíveis**, sistema deve selecionar dentre as páginas qual deverá ser liberada.
- **Algoritmos de substituição de página** tem objetivo de selecionar frames que tem menor chance de ser referenciado no futuro próximo.

SOBRE O OBJETIVO DOS ALGORITMOS...

- Selecionar por páginas ou frames que tem menor chance de ser referenciado futuramente, isso o tornará eficiente.
- Caso contrário, o frame poderá retornar diversas vezes para a RAM, gerando vários page faults e acesso à memória secundária.
- Melhor algoritmo? Aquele que conseguir escolher os frames que não utilizado no futuro ou levasse maior tempo.
- Cuidado aos algoritmos: Maior a complexidade --- Maior overhead (custo) para o S.O. processar, ou seja, gastar tempo sendo eficiente.

OBJETIVO PRINCIPAL DOS ALGORITMOS:

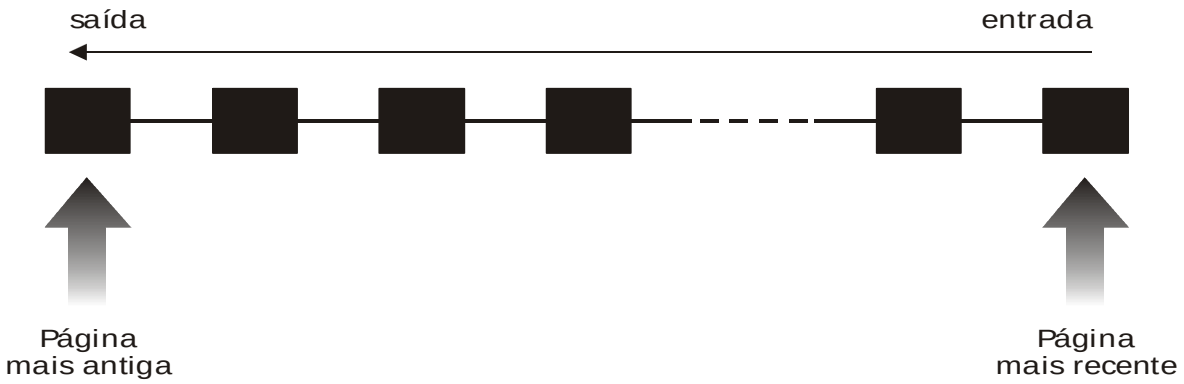
- MANTER O WORKING SET DOS PROCESSOS NA MEMÓRIA PRINCIPAL E AO MESMO TEMPO NÃO COMPROMETER O DESEMPENHO DO S.O.

ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINAS.

- FIFO (First-in-First-Out).
- ÓTIMO.
- ALEATÓRIO.
- LRU (Least-Recently-Used).
- LFU (Least-Frequently-Used).
- PRIORIDADE
- FIFO CIRCULAR (CLOCK) – 2° CHANCE.

FIFO

- A pagina que primeiro foi utilizada será a primeira a ser escolhida.
- Seleciona a paginas que esta há mais tempo na memória principal.
- O algoritmo deve ser implementado associando a cada pagina o momento em que foi carregada para a memória ou utilizando estrutura de fila.
- Baixo custo e de fácil implementação que consiste em substituir a página que foi **carregada** há mais tempo na memória (*a primeira página a entrar é a primeira a sair*).



EXEMPLO FIFO.

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 612314153414323

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

Fifo

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	5	1	1	1	1	1
F.2		1	1	1	1	4	4	4	3	3	3	3	3	2	2
F.3			2	2	2	2	1	1	1	4	4	4	4	4	3
PAGE FAULT	S	S	S	S	-	S	S	S	S	S	S	-	-	S	S

PAGE FAULT TOTAL= 12

ÓTIMO

- Seleciona para substituição a pagina que não será mais referenciada no futuro ou que levara maior intervalo de tempo para ser novamente utilizada.
- Impossível ser implementada pois o sistema não tem como conhecer o comportamento futuro das aplicações.
- Estratégia utilizada apenas como modelo comparativo.

ÓTIMO

- É o que apresenta o melhor desempenho computacional e o que minimiza o número de faltas de páginas. No entanto, sua implementação é praticamente impossível.
- A ideia do algoritmo é retirar da memória a página que vai demorar mais tempo para ser referenciada novamente. Para isso, **o algoritmo precisaria saber, antecipadamente, todos os acessos à memória realizados pela aplicação**, o que é ***impossível*** em um caso real. Por estes motivos, o algoritmo ótimo só é utilizado em simulações para se estabelecer o valor ótimo e analisar a eficiência de outras propostas elaboradas.

EXEMPLO ÓTIMO

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 612314153414323

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

Ótimo

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	3	3	3	3	3	3	3	3
F.2		1	1	1	1	1	1	5	5	5	1	1	1	2	2
F.3			2	2	2	4	4	4	4	4	4	4	4	4	4
PAGE FAULT	S	S	S	S	-	S	-	S	-	-	S	-	-	S	-

RAM

F.1	3
F.2	2
F.3	4

PAGE FAULT TOTAL= 8

ALEATÓRIO

- Algoritmo não utiliza nenhum critério de seleção.
- Todas as páginas tem mesma chance de serem selecionadas.
- Estratégia que seleciona através de sorteio aleatório qual página será substituída.
- Apesar de ser uma estratégia que consome poucos recursos, tem baix eficiência.

LFU (LEAST-FREQUENTLY-USED)

- Seleciona a pagina menos referenciada, ou o frame menos utilizado.
- **Freqüência menos utilizada.**
- Mantem um contador com números de referencia para cada pagina na memória principal.
- A pagina que possuir contador menor numero será escolhida.
- Algoritmo busca manter na memória principal páginas com bastante utilização.

EXEMPLO LFU

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 612314153414323

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

LFU

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3	C
F.1	6	6	6	3	3	3	3	5	5	4	4	4	4	4	4	5
F.2		1	1	1	1	1	1	1	1	1	1	1	1	2	2	5
F.3			2	2	2	4	4	4	3	3	3	3	3	3	3	5
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-	-	S	-	PAGE FAULT TOTAL=

RAM

F.1	4
F.2	2
F.3	3

LRU (LEAST-RECENTLY-USED)

- Seleciona a pagina que esta há mais tempo sem ser referenciada.
- **Recentemente menos utilizada.**
- Para implementar este algoritmo é necessário que cada pagina tenha associado o momento do seu ultimo acesso.
- É permitido implementação através de lista encadeada, onde todas as páginas estariam ordenadas pelo momento da última referencia.

EXEMPLO LRU

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 612314153414323

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

LRU

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	5	1	1	1	2	2
F.2		1	1	1	1	1	1	1	1	4	4	4	4	4	4
F.3			2	2	2	4	4	4	3	3	3	3	3	3	3
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	S	-	-	S	-

RAM

F.1	2
F.2	4
F.3	3

PAGE FAULT TOTAL= 10

PRIORIDADE.

- Seleciona para substituição um bloco com um processo de **menor prioridade**
- Quando prioridade igual escolhe o primeiro frame (mais próximo).
- Princípio: "quem tem mais prioridade tem mais chance de ficar alocado".

EXEMPLO PRIORIDADE

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 6123141534143

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

1 - Prioridade 3
2 - Prioridade 2
3 - Prioridade 1
4 - Prioridade 0
5 - Prioridade 0
6 - Prioridade 1

PRIORIDADE

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	4	4	5	3	4	4	4	3	3	3
F.2		1	1	1	1	1	1	1	1	1	1	1	1	1	1
F.3			2	2	2	2	2	2	2	2	2	2	2	2	2
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-	S	-	-

RAM

F.1	3
F.2	1
F.3	2

PAGE FAULT TOTAL= 9

FIFO CIRCULAR (CLOCK)

- Utiliza como base o FIFO.
- Páginas alocadas estão em uma estrutura de lista circular, semelhante a um relógio.
- Cada página possui um bit de referencia desligado (0) ou ligado (1).
- Bit de referencia desligado ou ligado representa se a página foi utilizada recentemente.

FIFO CIRCULAR (CLOCK).

- Cada pagina tem um Byte R referenciado antes de remover a pagina + antiga (igual FIFO) o byte é verificado.
- Se R= 0 troca a pagina Se r=1 pagina vai para o fim da fila como se houvesse chegado agora, muda R para 0.
- Se todos R=1 da uma volta completa e troca a ultima pagina.
- Passa R para 1 quando não existe falta de pagina.

EXEMPLO FIFO CLOCK

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (1,2,3)

P2: (4,5)

P3: (6)

String Referência: 6123141534143

Frames memoria Ram: 3

Paginas na memória secundária (Ex. de secundária 150% principal):

1	2	3	4	5	6		
F.1	F.2	F.3	F.4	F.5	F.6	F.7	F.8

3 Páginas de Ram

+ 5 paginas na secudária

Total de 8 Paginas.

FIFO CIRCULAR (CLOCK)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3											
F.2		1	1	1											
F.3			2	2											
PAGE FAULT	S	S	S	S											

R	
F.1	0
F.2	0
F.3	0

PAGE FAULT TOTAL=

FILA

String Referência: 612314153414323

Frame	1	2	3	1											

EXEMPLO
FIFO CLOCK (2)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3										
F.2		1	1	1	1										
F.3			2	2	2										
PAGE FAULT	S	S	S	S	-										

	R
F.1	0
F.2	1
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1											

EXEMPLO
FIFO CLOCK (3)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3									
F.2		1	1	1	1	1									
F.3			2	2	2	4									
PAGE FAULT	S	S	S	S	-	S									

	R
F.1	0
F.2	0
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3									

Passou a vez do 2, colocando em 1º lugar mas não houve subst.

Trocou o próximo, frame 3. Zerar o byte do Frame 2.

EXEMPLO
FIFO CLOCK (4)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3								
F.2		1	1	1	1	1	1								
F.3			2	2	2	4	4								
PAGE FAULT	S	S	S	S	-	S	-								

	R
F.1	0
F.2	1
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3									

EXEMPLO
FIFO CLOCK (5)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5							
F.2		1	1	1	1	1	1	1							
F.3			2	2	2	4	4	4							
PAGE FAULT	S	S	S	S	-	S	-	S							

	R
F.1	0
F.2	1
F.3	0

FILA

String Referência: 6123141534143 23

Frame	1	2	3	1	2	3	1								

EXEMPLO
FIFO CLOCK (6)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5						
F.2		1	1	1	1	1	1	1	1						
F.3			2	2	2	4	4	4	3						
PAGE FAULT	S	S	S	S	-	S	-	S	S						

R	
F.1	0
F.2	0
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3	1	2	3						

Passou a vez do 2, colocando em 1º lugar mas não houve subst.

Trocou o próximo, frame 3. Zerar o byte do Frame 2.

EXEMPLO
FIFO CLOCK (7)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	4					
F.2		1	1	1	1	1	1	1	1	1					
F.3			2	2	2	4	4	4	3	3					
PAGE FAULT	S	S	S	S	-	S	-	S	S	S					

R	
F.1	0
F.2	0
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3	1	2	3	1					

EXEMPLO
FIFO CLOCK (8)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	4	4				
F.2		1	1	1	1	1	1	1	1	1	1				
F.3			2	2	2	4	4	4	3	3	3				
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-				

	R
F.1	0
F.2	1
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3	1	2	3	1					

EXEMPLO
FIFO CLOCK (9)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	4	4	4			
F.2		1	1	1	1	1	1	1	1	1	1	1			
F.3			2	2	2	4	4	4	3	3	3	3			
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-			

R	
F.1	1
F.2	1
F.3	0

FILA

String Referência: 6123141534143 23

Frame	1	2	3	1	2	3	1	2	3	1					

EXEMPLO
FIFO CLOCK (10)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	6	3	3	3	3	5	5	4	4	4	4	
F.2		1	1	1	1	1	1	1	1	1	1	1	1	1	
F.3			2	2	2	4	4	4	3	3	3	3	3	3	
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-	-		

	R
F.1	1
F.2	1
F.3	1

FILA

String Referência: 6123141534143 23

Frame	1	2	3	1	2	3	1	2	3	1					

EXEMPLO
FIFO CLOCK (11)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	6	3	3	3	5	5	4	4	4	4	4	
F.2		1	1	1	1	1	1	1	1	1	1	1	1	2	
F.3			2	2	2	4	4	4	3	3	3	3	3	3	
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-	-	S	

	R
F.1	0
F.2	0
F.3	0

FILA

String Referência: 612314153414323

Frame	1	2	3	1	2	3	1	2	3	1	2	3	1	2	

Passou por todos oferecendo 2º chance e zerando o byte.

Como todos com byte 1 voltou no primeiro, frame 2.

EXEMPLO
FIFO CLOCK (12)

Ref.	6	1	2	3	1	4	1	5	3	4	1	4	3	2	3
F.1	6	6	6	3	3	3	3	5	5	4	4	4	4	4	4
F.2		1	1	1	1	1	1	1	1	1	1	1	1	2	2
F.3			2	2	2	4	4	4	3	3	3	3	3	3	3
PAGE FAULT	S	S	S	S	-	S	-	S	S	S	-	-	-	S	-

	R
F.1	0
F.2	0
F.3	1

FILA

String Referência: 6123141534143 23

Frame	1	2	3	1	2	3	1	2	3	1	2	3	1	2	

EXERCÍCIOS.

- Representar os algoritmos de substituição de páginas para os processos:

Prioridade	
0	1
1	2
2	0
3	0
5	1
6	1
8	2

Processos P1, P2 e P3, que utilizam pages de memória, respectivamente.

Exemplo P1 esta dividido e utilizando 3 paginas de mémoria.

P1: (0,1,2,3)

P2: (4)

P3: (5,6)

P4: (8)

String Referência: 13653852:

Frames memoria Ram: 5

Paginas na memória secundária (Ex. de secundária 150% principal):

5 Páginas de Ram

+ 7 paginas na secudária

Total de 12 Paginas.

REFERÊNCIA BIBLIOGRÁFICA.

- TANEMBAUM, Andrew S. Sistemas Operacionais Modernos. 2° Ed. Pearson, 2005.
- MACHADO, Francis Berenger; MAIA, Luiz Paulo. Arquitetura de Sistemas Operacionais. 4° Edição, LTC, 1996.