



# Alocação de Memória

ARTHUR SALIM DA COSTA

LAIS SILVA COSTA

MARIA HELENA DE SOUSA COSTA

# Conteúdo

---

01 Alocação Contígua

02 Alocação Particionada Estática

03 Alocação Particionada Dinâmica

04 Overlay

05 Swapping

06 Memória Virtual - Paginação

07 Algoritmo NRU

08 Algoritmo FIFO

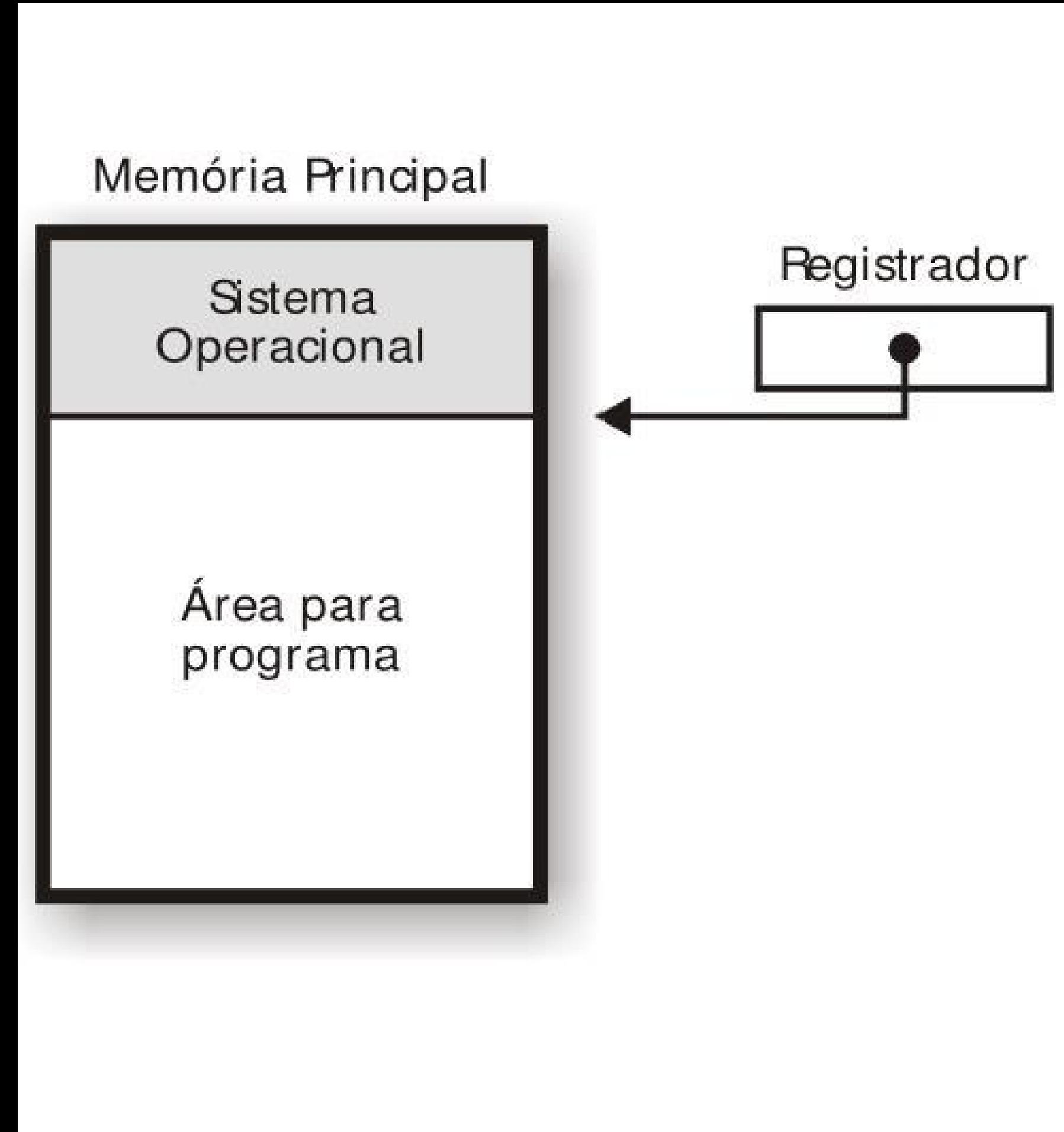
09 Algoritmo do Relogio

10 Algoritmo LFU

11 Algoritmo LRU

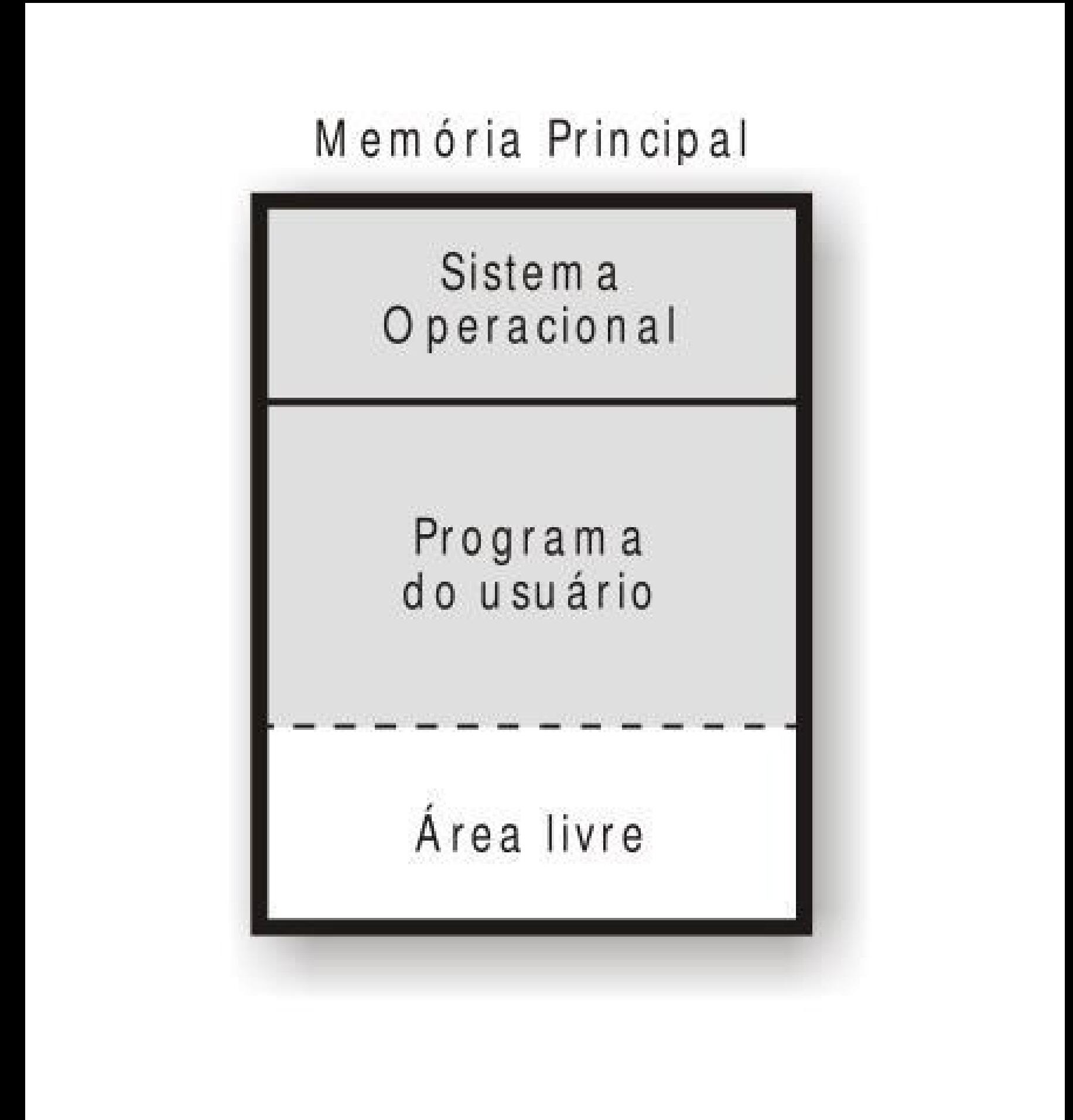
# Alocação Contígua

- Memória principal é subdividida em duas áreas
- programa é alocado em um único bloco contíguo de memória
- o programador deve desenvolver suas aplicações preocupado, apenas, em não ultrapassar o espaço de memória disponível



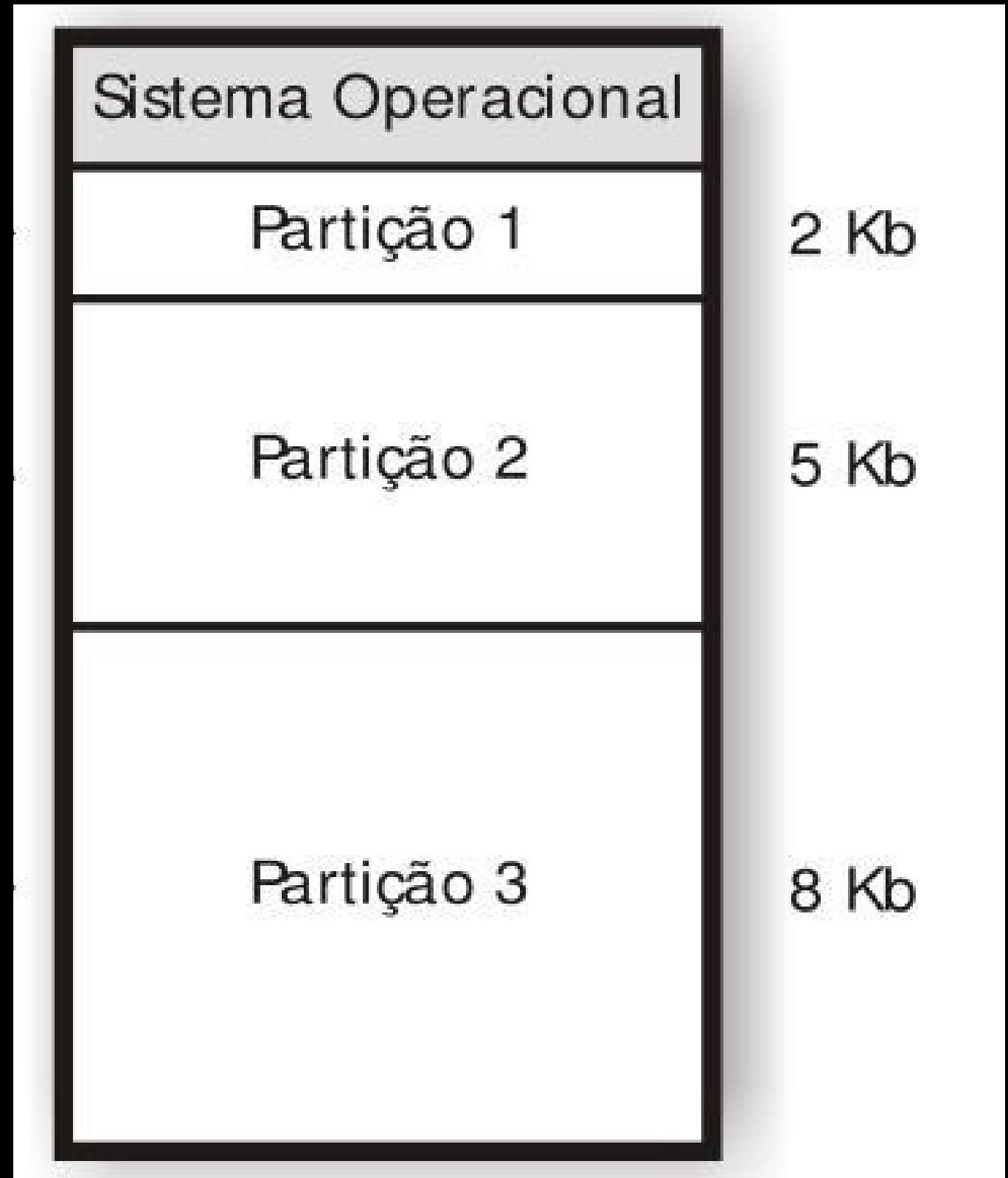
# Alocação Contígua

- Memória principal é subdividida em duas áreas
- programa é alocado em um único bloco contíguo de memória
- o programador deve desenvolver suas aplicações preocupado, apenas, em não ultrapassar o espaço de memória disponível



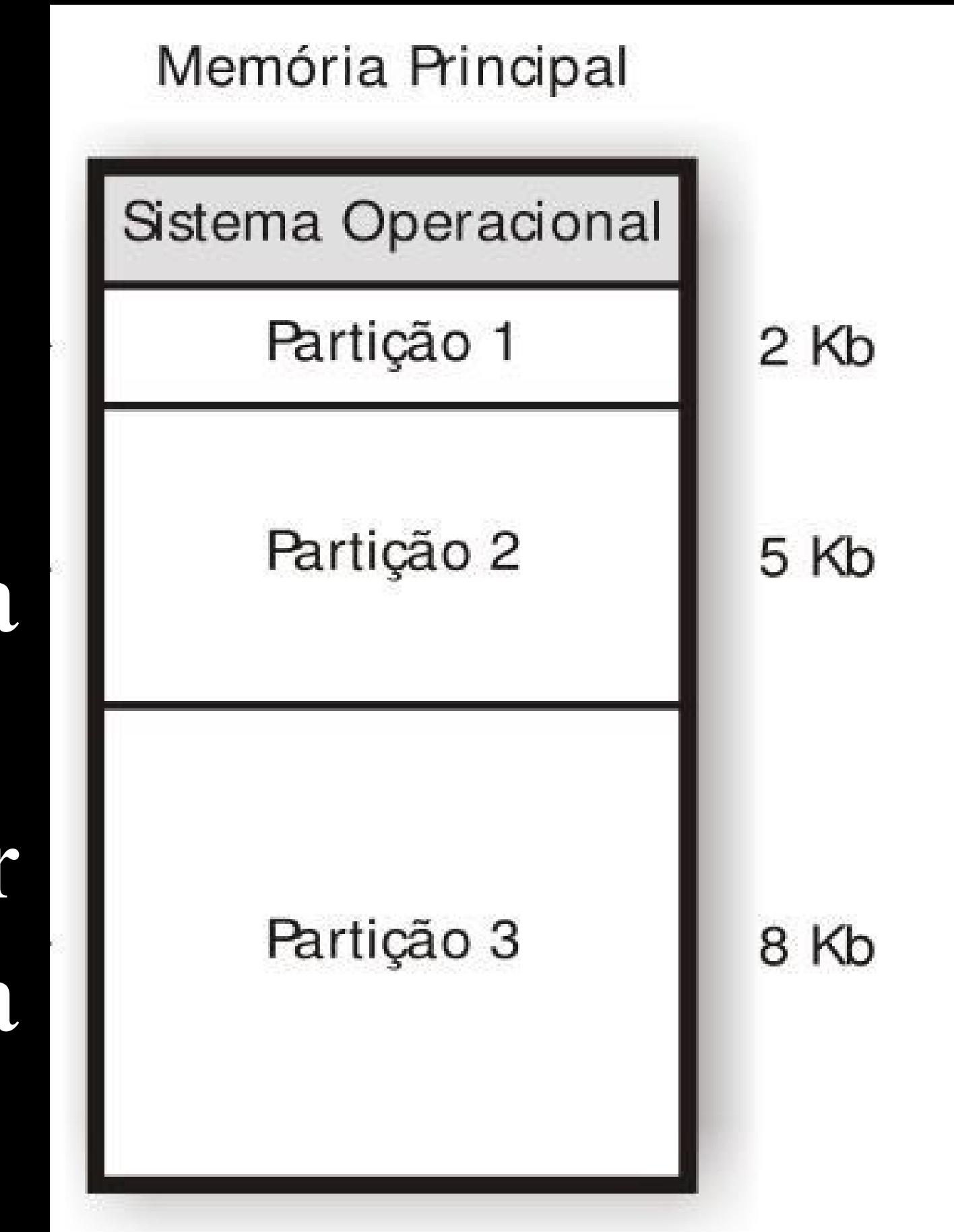
# Alocação Particionada

- Alocação particionada estática
  - Absoluta
  - Reallocável
- Alocação particionada dinâmica

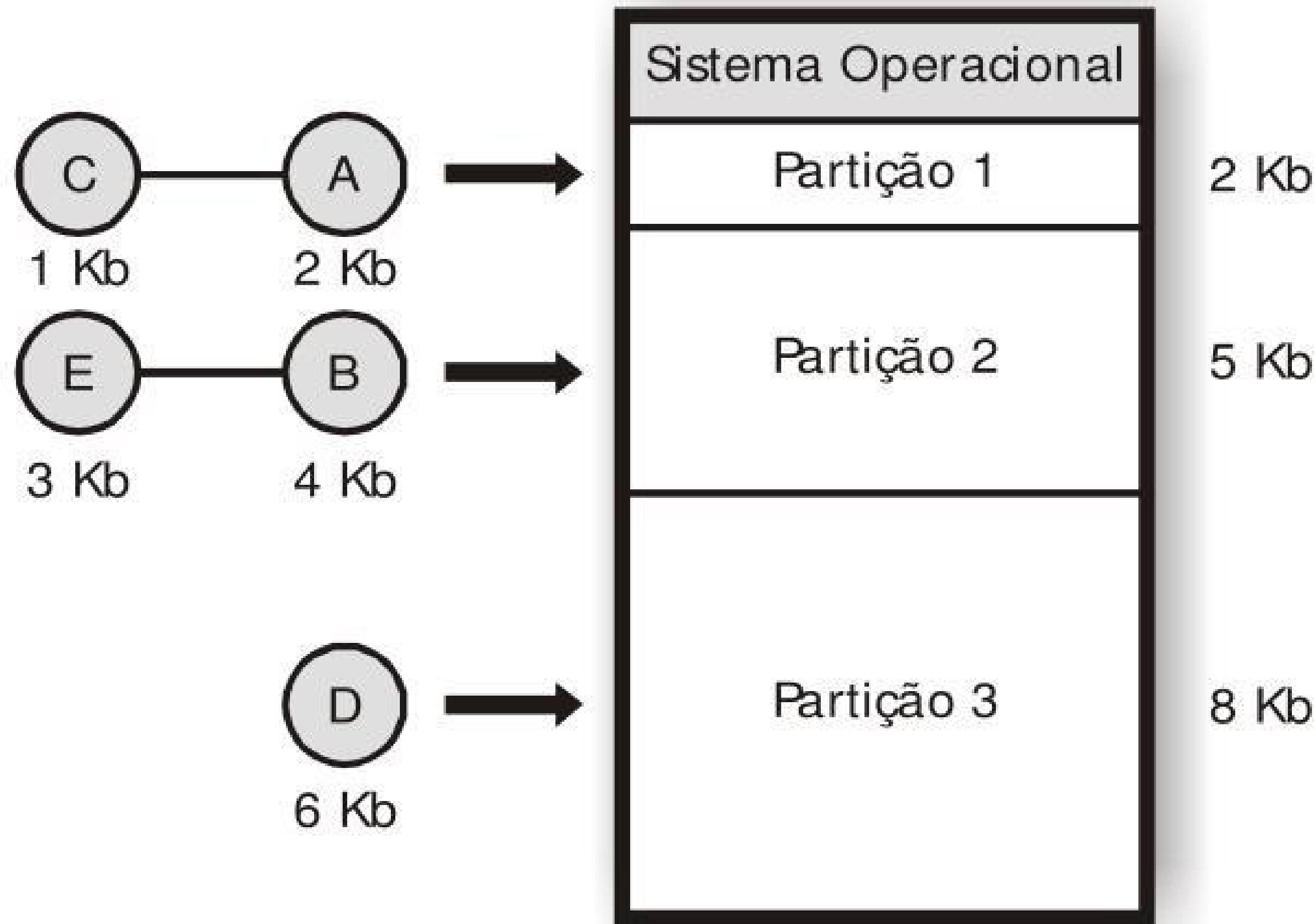


# Alocação Particionada Estática Absoluta

- Código absoluto
- Programas exclusivos para partições específicas.
- Cada processo só poderia ser executados em uma mesma partição



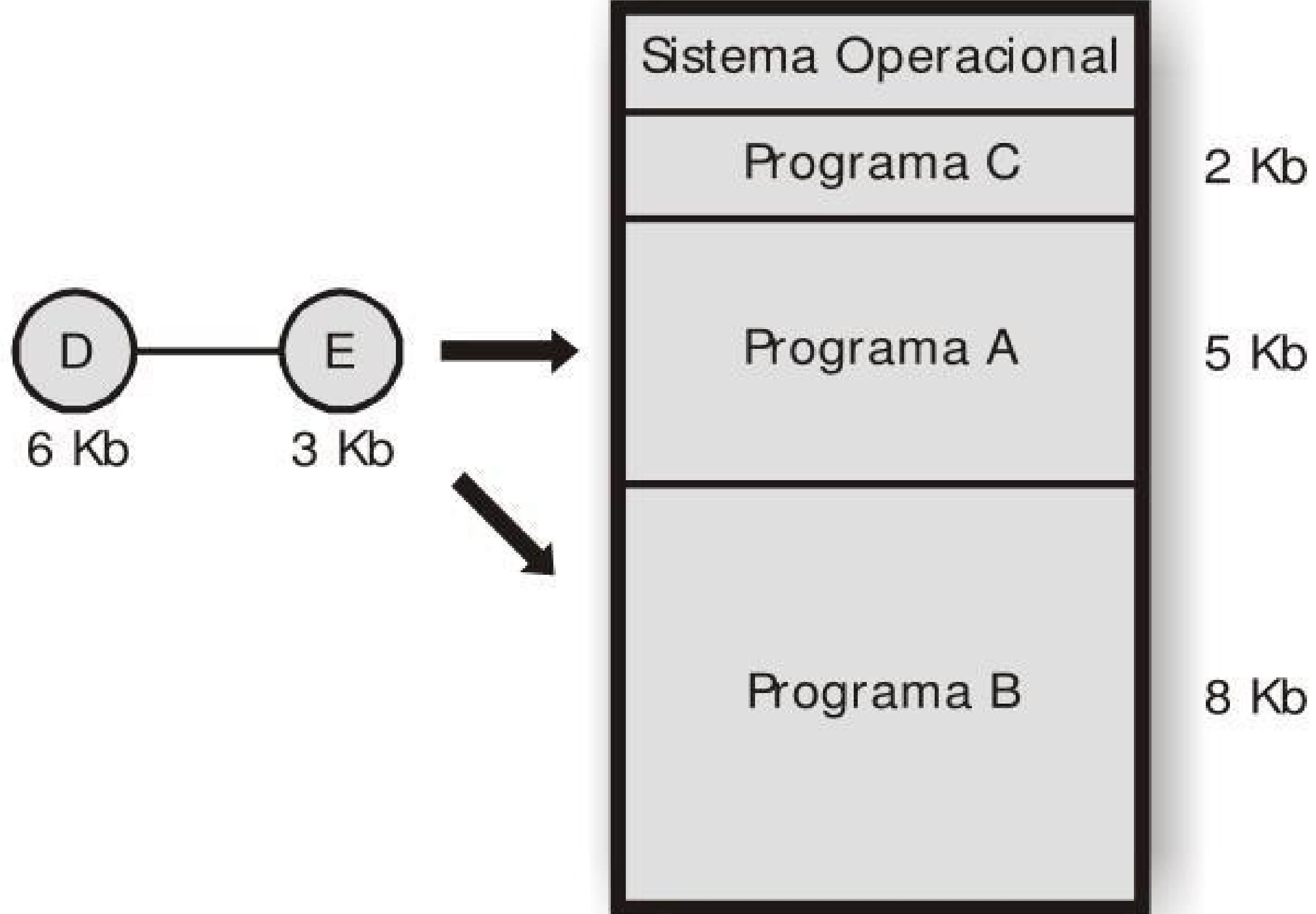
# Memória Principal



# Alocação Particionada Estática Reallocável

- Código reallocável
- Programas podem rodar em qualquer partição

# Memória Principal



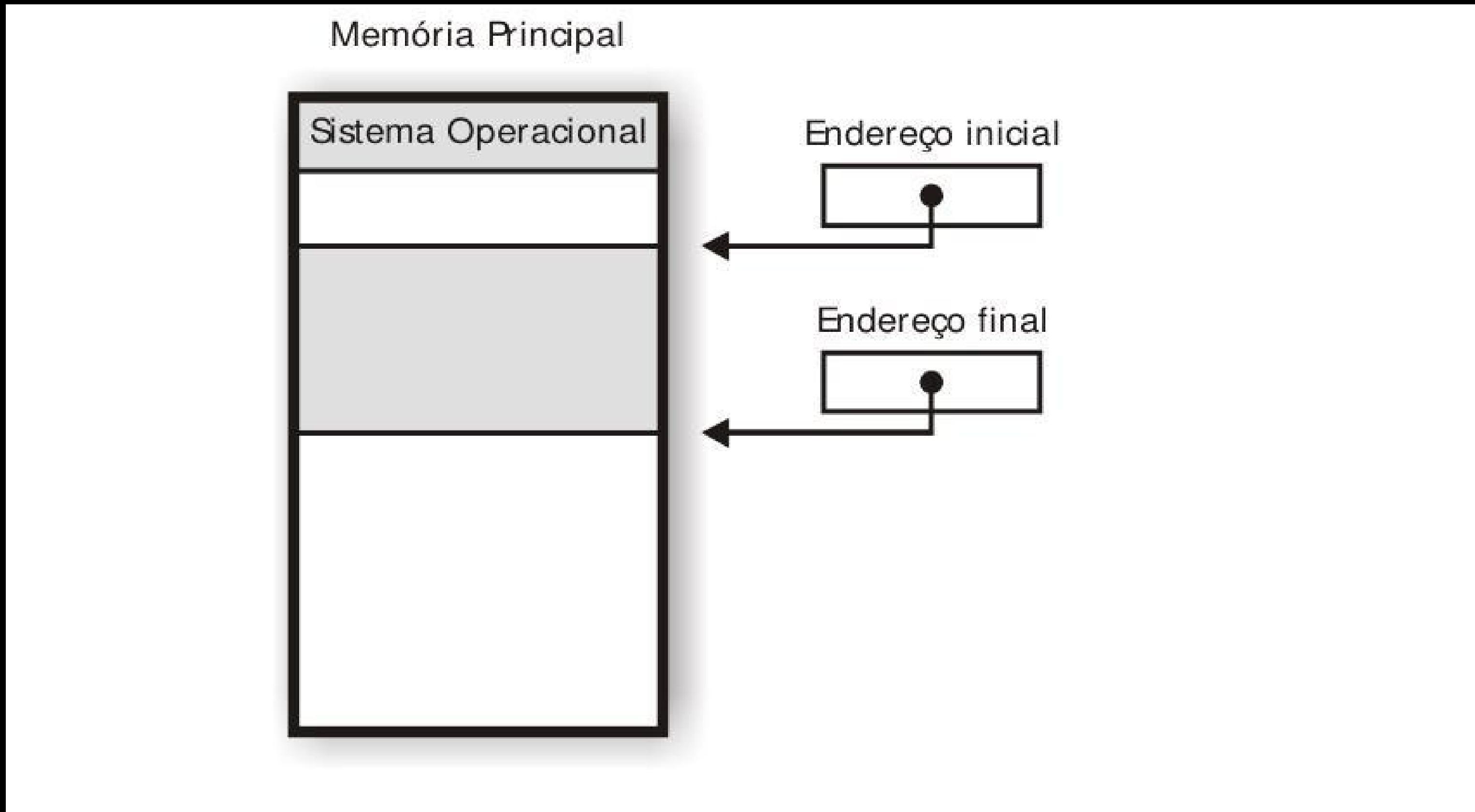
# Tabela de Alocação de Partições

Partição	Tamanho	Livre
1	2 Kb	Não
2	5 Kb	Sim
3	8 Kb	Não

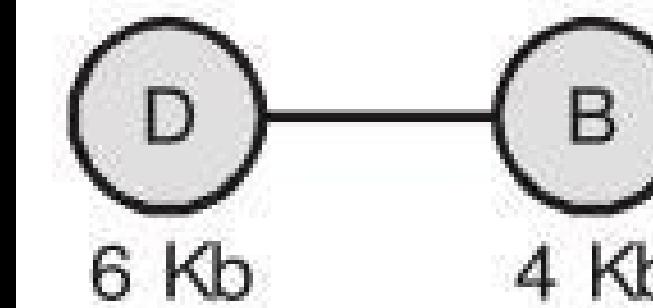
Memória Principal



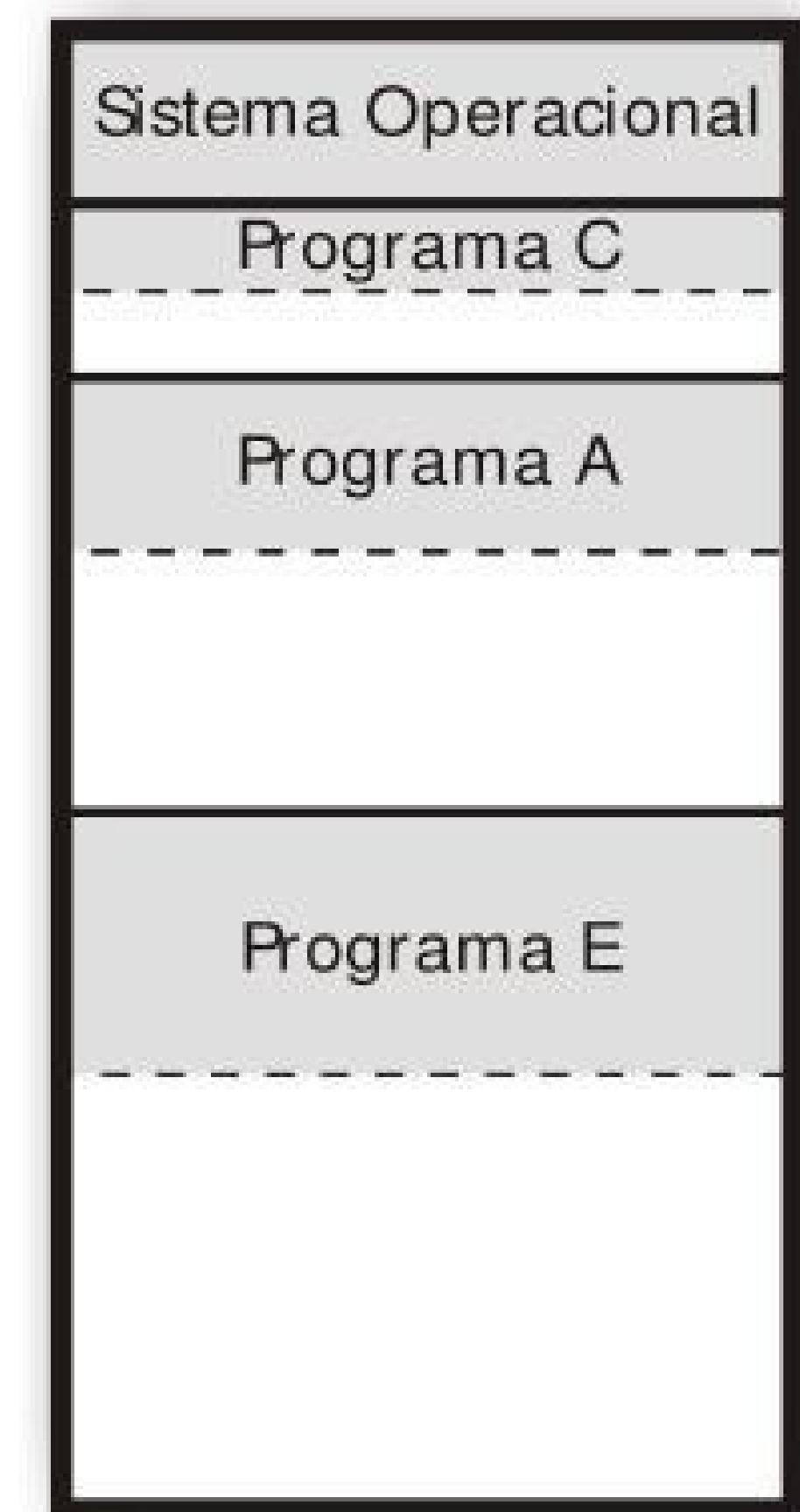
# Proteção na Alocação Particionada



# Fragmentação Interna



Memória Principal



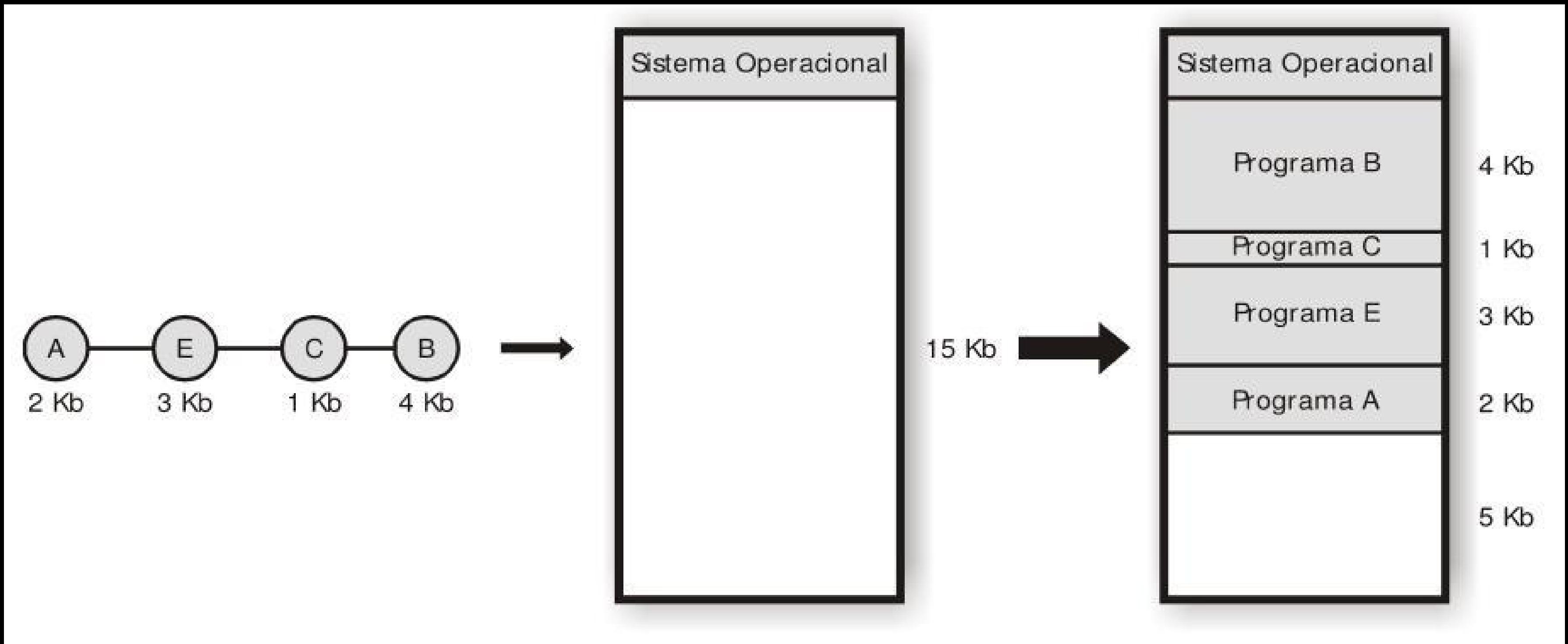
1 Kb

3 Kb

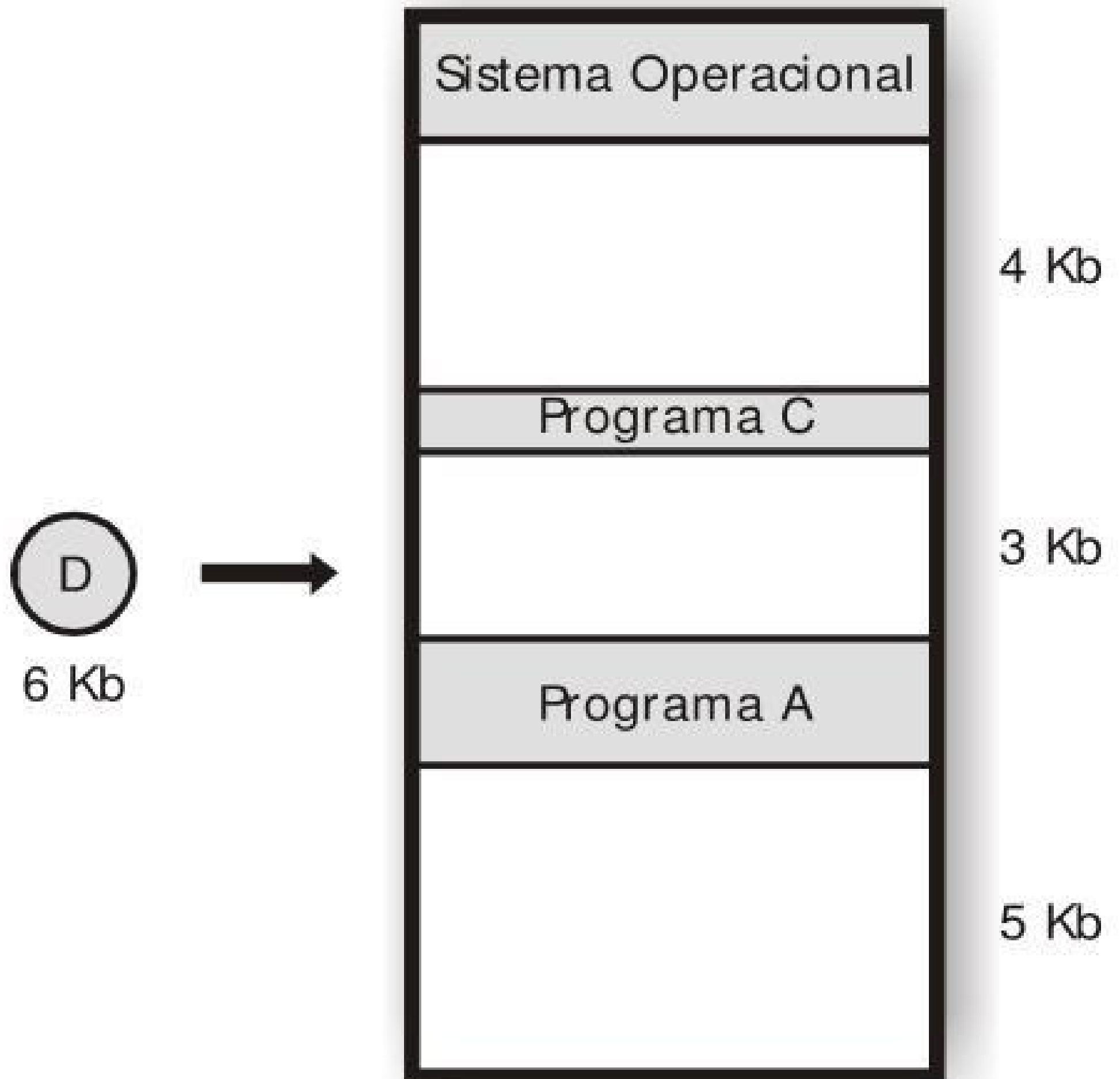
5 Kb

# Alocação Particionada Dinâmica

- Não há partições de tamanho fixo
- Não ocorre fragmentação interna

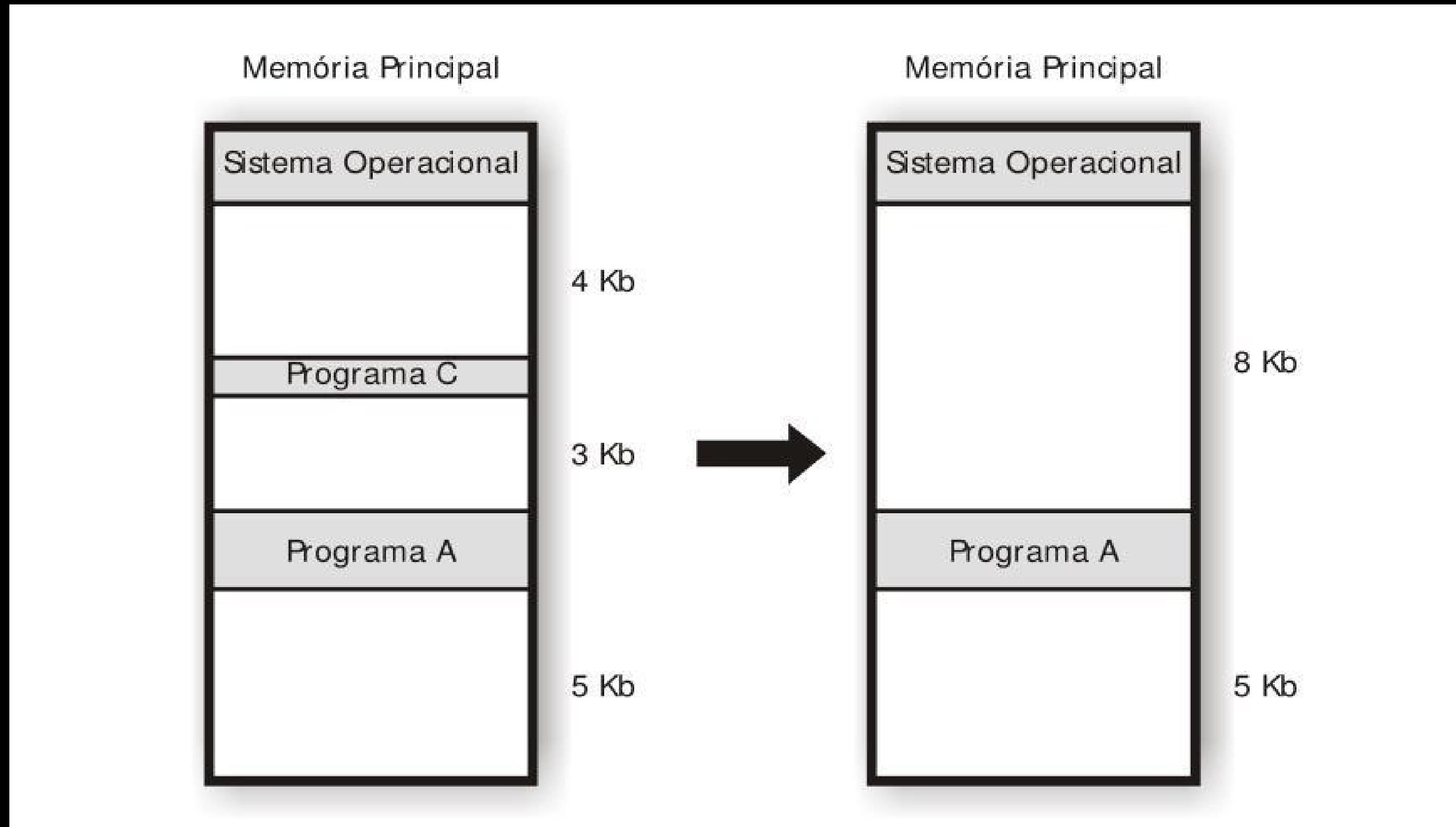


## Memória Principal

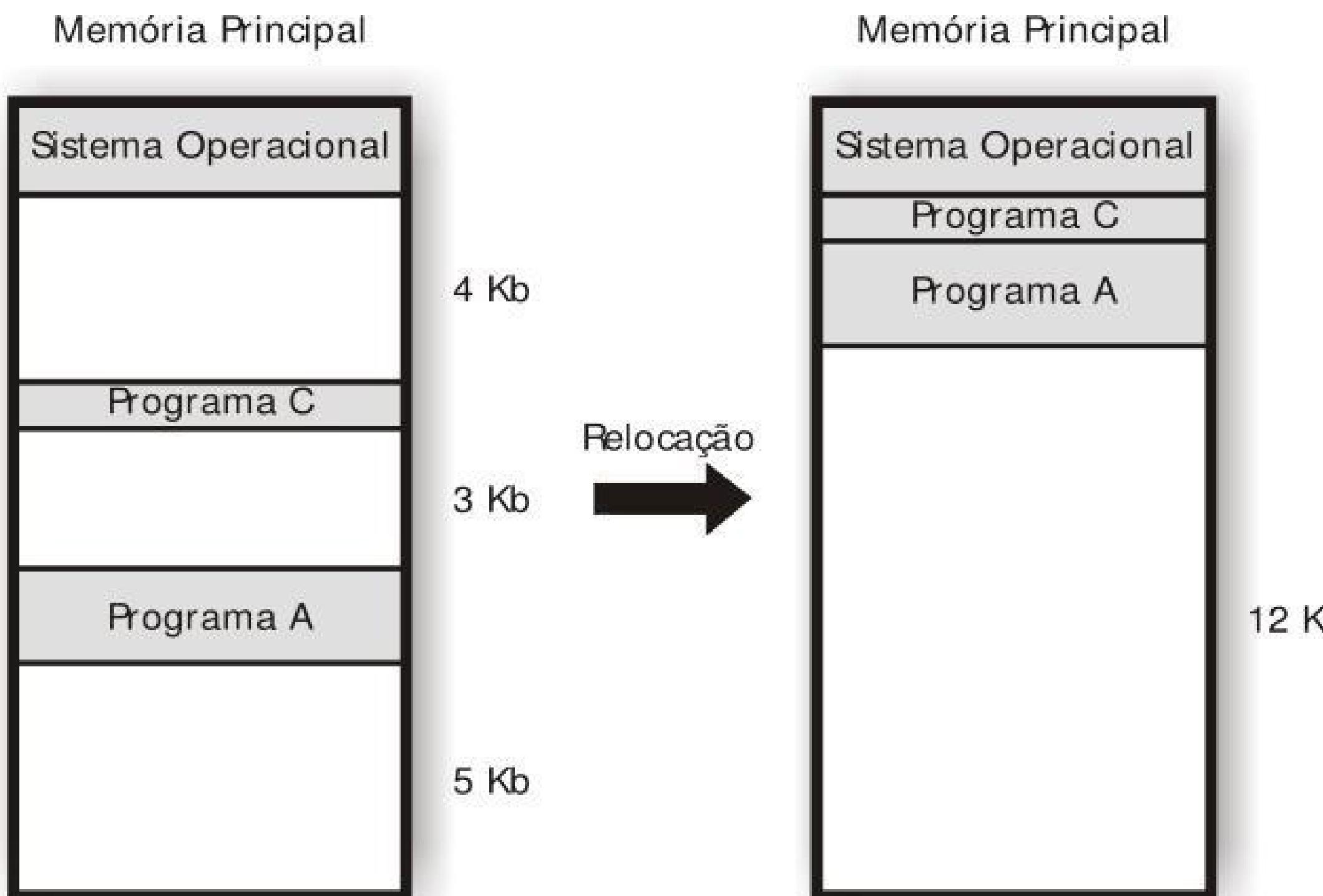


# Fragmentação Externa

# Solução 1 Para Fragmentação Interna

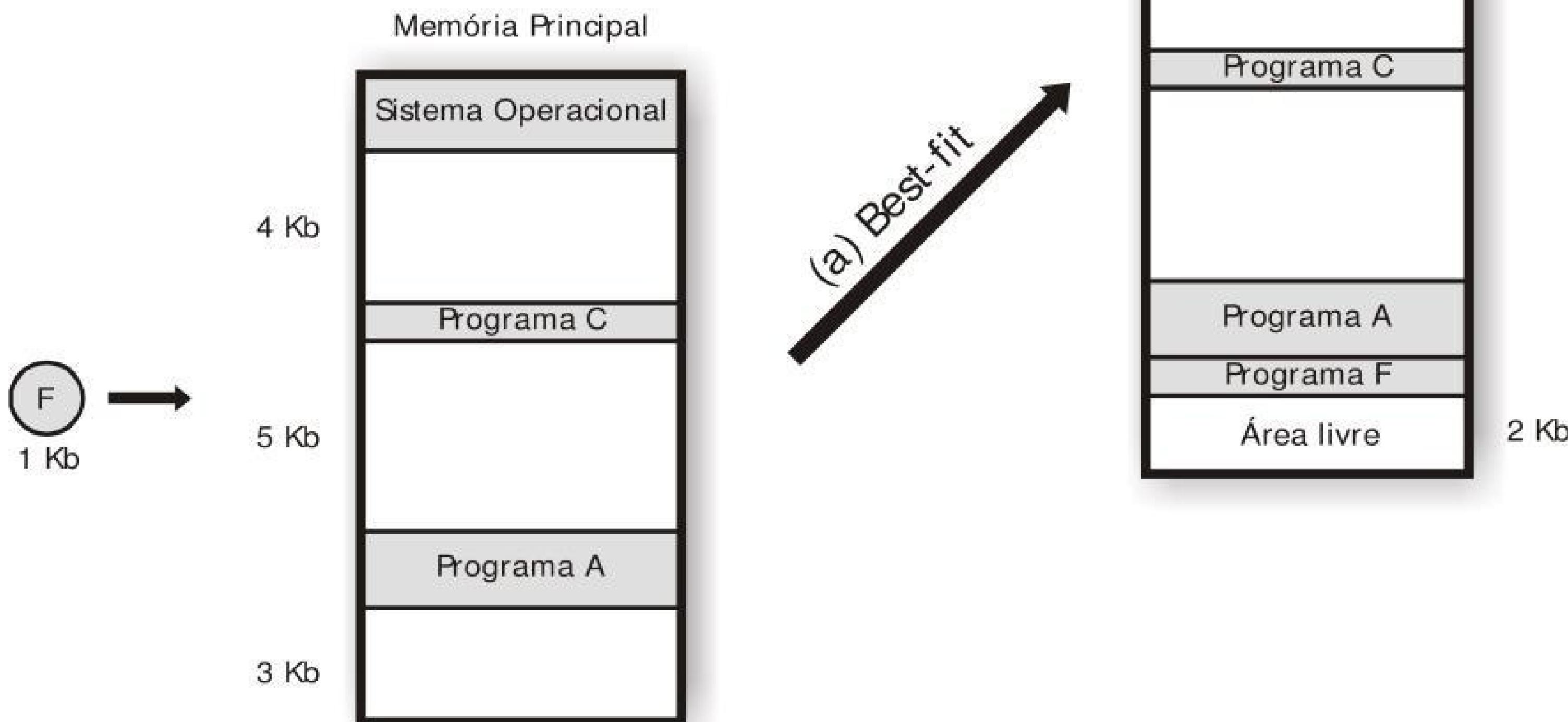


# Solução 2 Para Fragmentação Interna



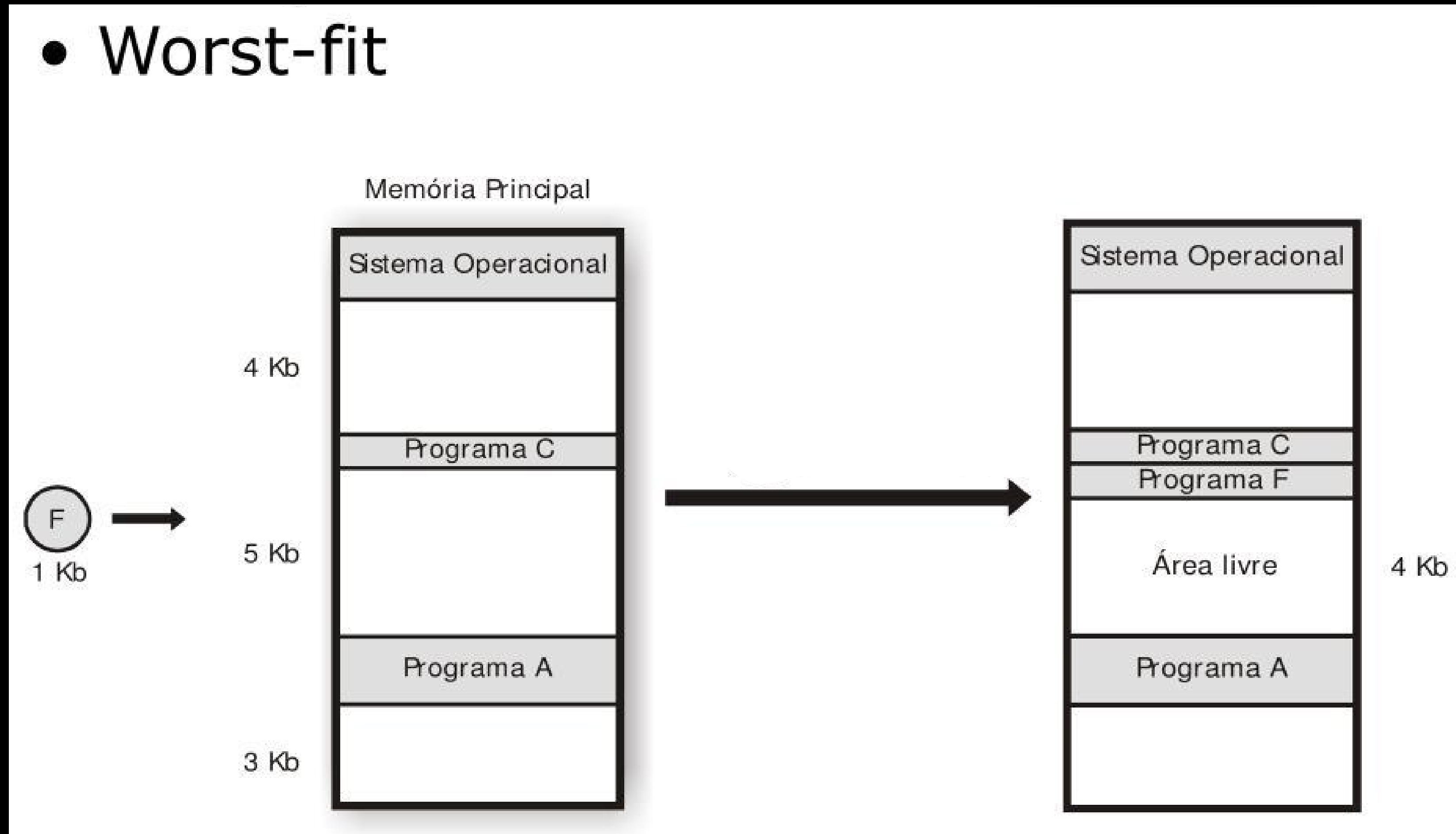
# Estratégias de Alocação de Partição

- Best-fit



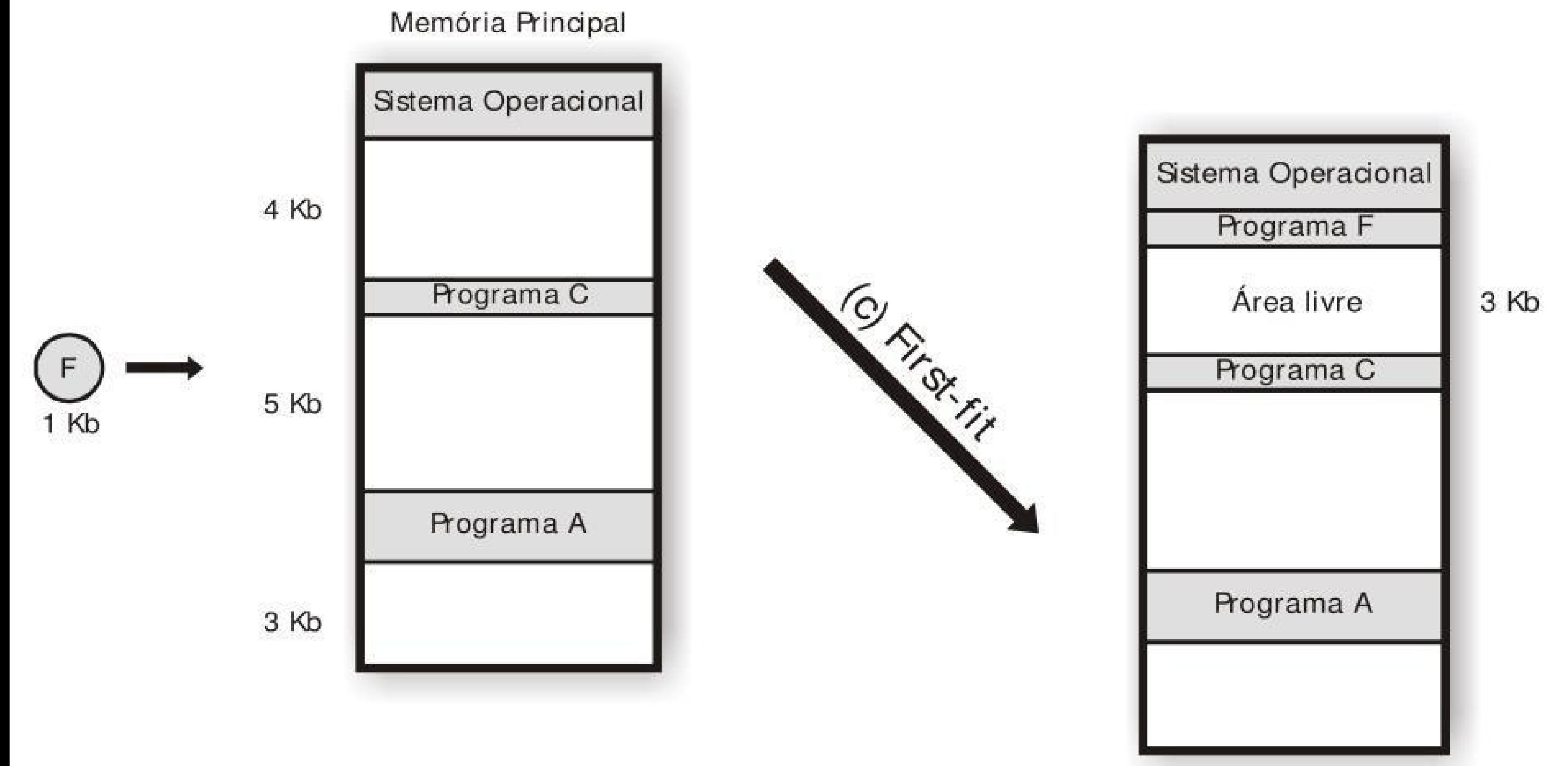
# Estratégias de Alocação de Partição

- Worst-fit



# Estratégias de Alocação de Partição

- First-fit



# OVERLAY

## DEFINIÇÃO

Os programas eram **limitados** ao tamanho da RAM disponível.

A solução foi **dividir** o programa em partes (**módulos**), que pudessem executar de forma **independentemente** um da outro utilizando a mesma área da memória.

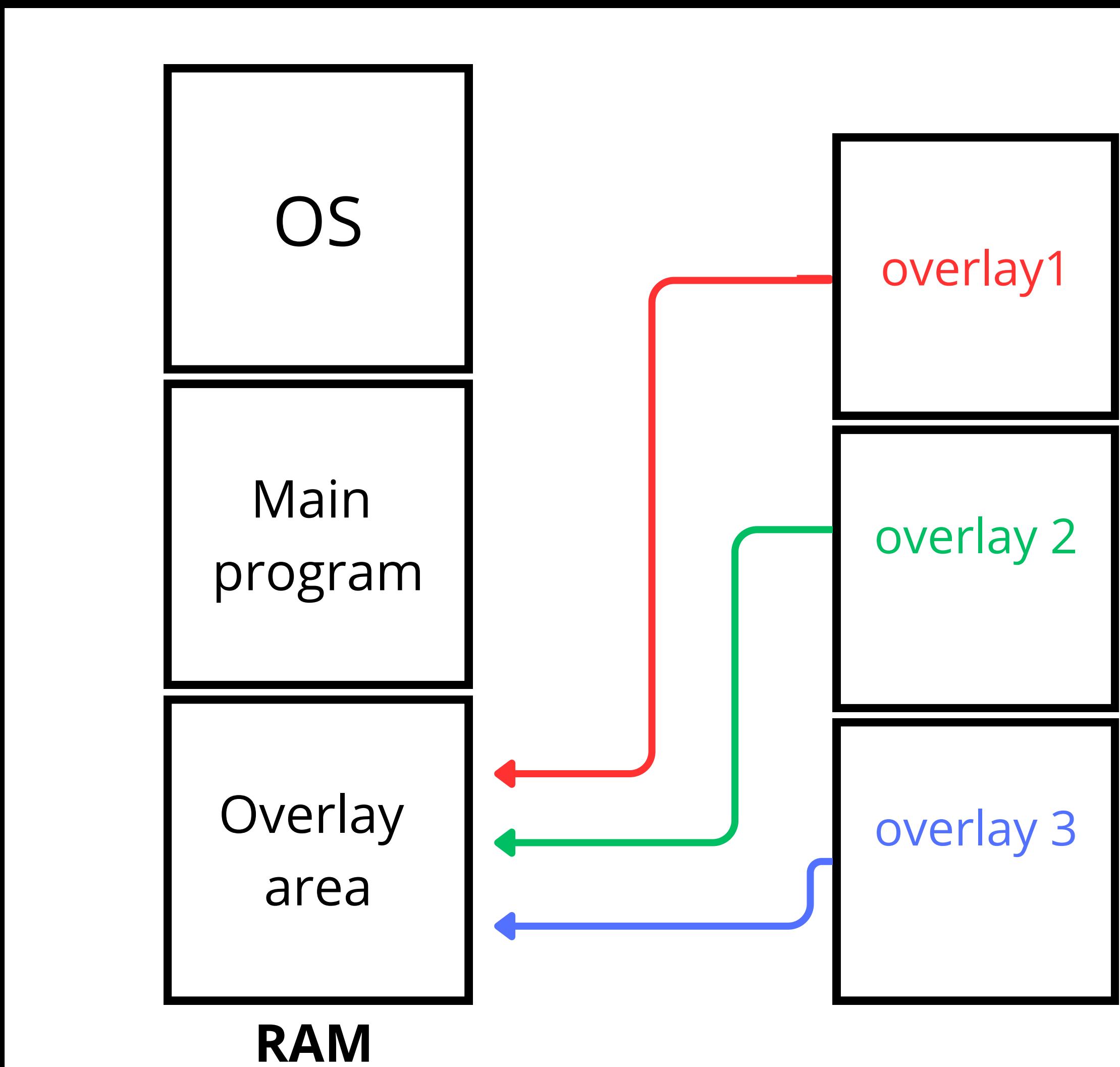
Esta técnica recebeu o nome de **overlay**.

## FUNCIONAMENTO

Apenas um módulo é carregado na RAM por vez.

Se o programa precisa de diferentes partes do código o S.O troca os módulos.

Na mesma região de memória sobrepondo a região de memória existente.



# Overlay

---

## Vantagens

- Execução de Programas Grandes
- Utilização Eficiente da Memória
- Compatibilidade com Restrições de Hardware

## Desvantagens

- Complexidade na Programação
  - Overhead de Troca de Overlays
  - Limitações de Tempo de Execução
-

# SWAPPING

## DEFINIÇÃO

Transferência temporária de programas da RAM para o armazenamento em **disco**.

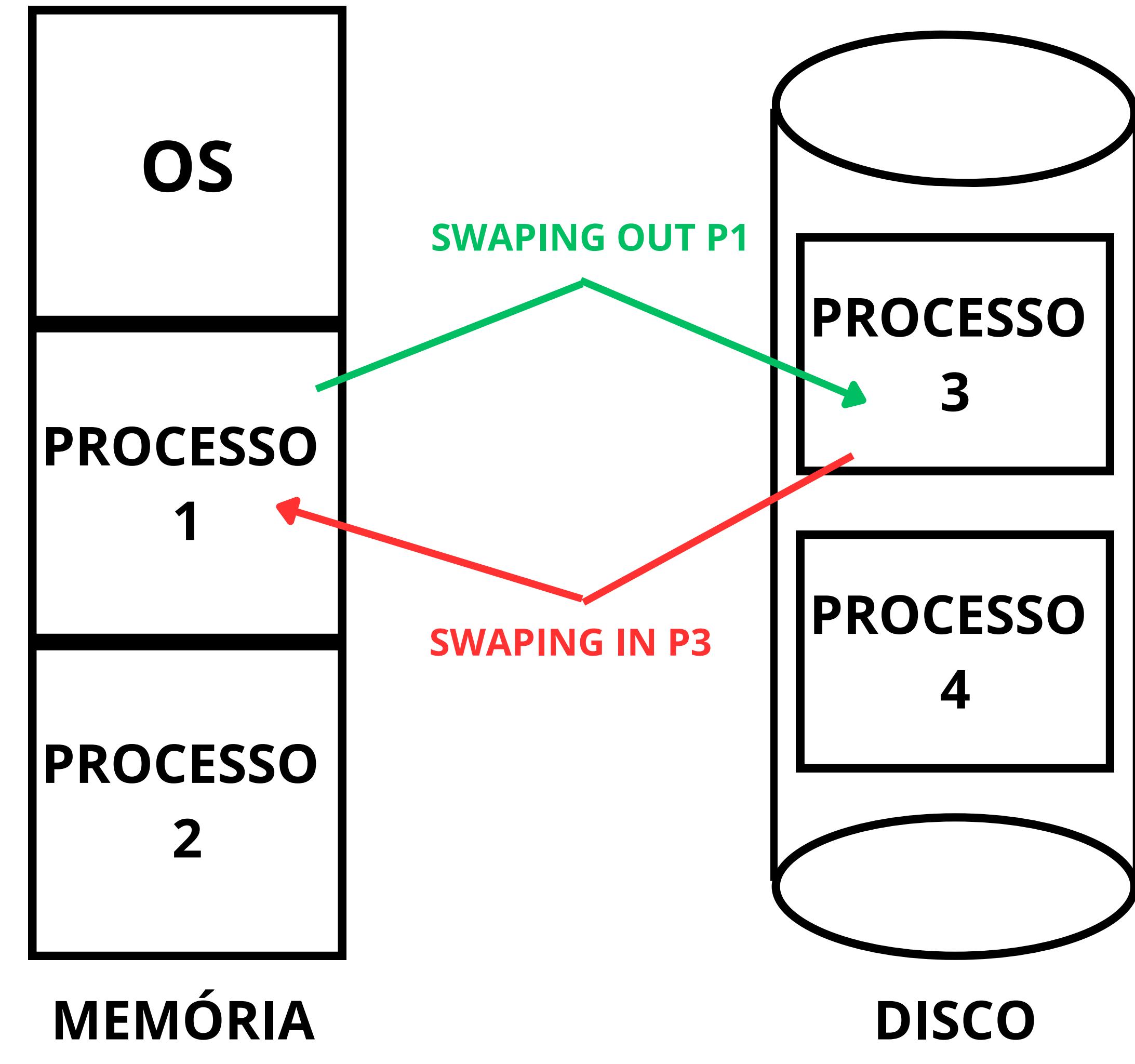
Tem por objetivo liberar espaço na RAM para **alocar novos** processos ou para dar mais **espaço** a processos ativos.

## FUNCIONAMENTO

Processo não utilizado é levado ao disco.

Quando esse processo para o disco precisar ser acessado novamente.

O S.O o move de volta da área de armazenamento para a RAM.



# Swapping

---

## Vantagens

- Capacidade Adicional
- Flexibilidade
- Adaptabilidade Dinâmica

## Desvantagens

- Desempenho
- I/O Intensivo
- Fragmentação do Disco



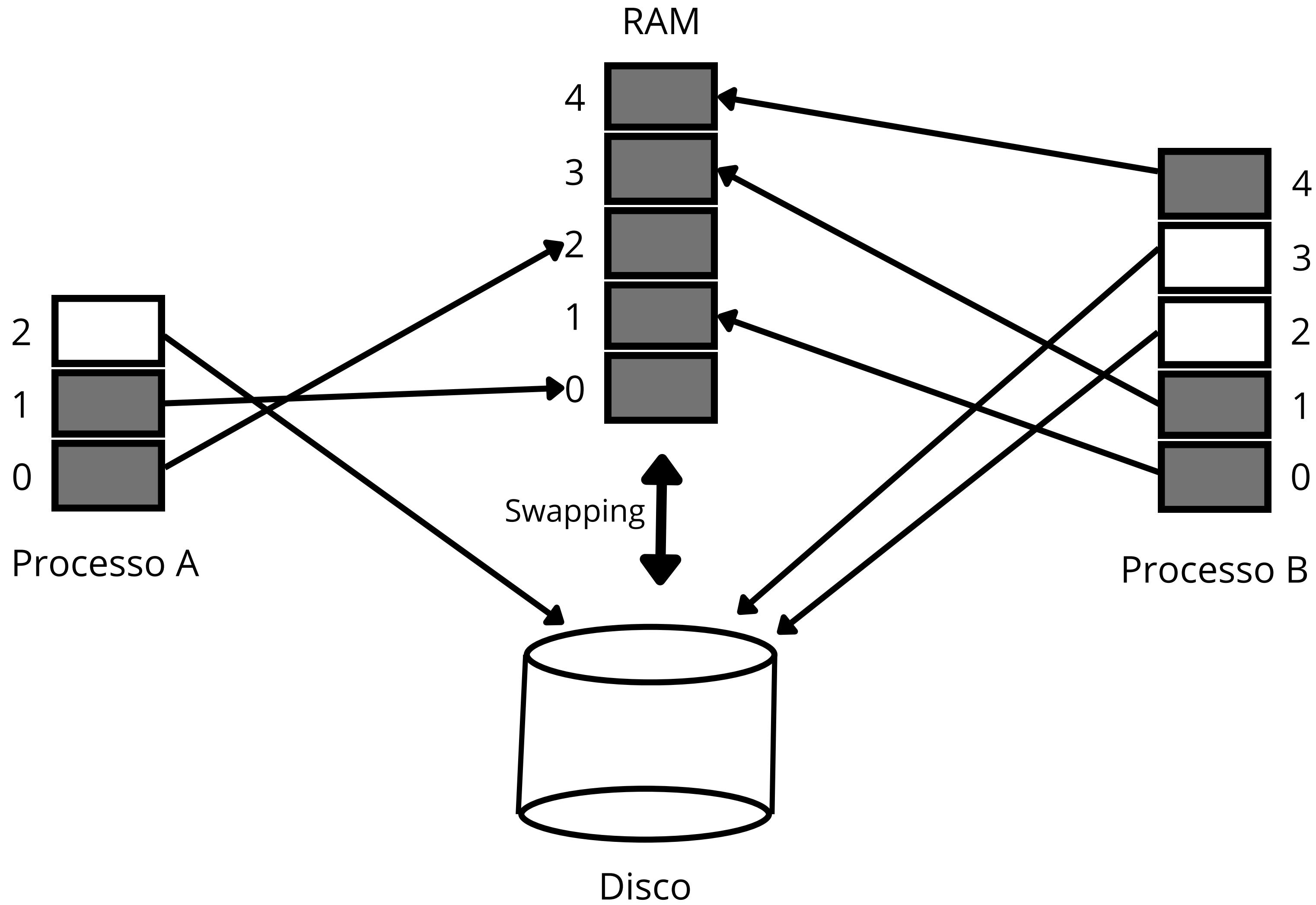
# MEMÓRIA VIRTUAL

## DEFINIÇÃO

Técnica que envolve o uso de uma combinação de memória física e espaço em disco para armazenar dados temporariamente.

Chamada de memória lógica pois não é real sendo apenas uma simulação da memória principal.

Consiste na virtualização de endereços físicos.





# MEMÓRIA VIRTUAL PAGINAÇÃO

## DEFINIÇÃO

Uma **técnica** utilizada para implementar a memória virtual.

**Divisão** da memória virtual e memória física em blocos chamados "páginas" e “moldura de páginas” respectivamente.

Esses blocos têm um **tamanho fixo**.

S.O mantém em memória uma **tabela de páginas** para cada processo que relaciona as duas.

# Fucionamento

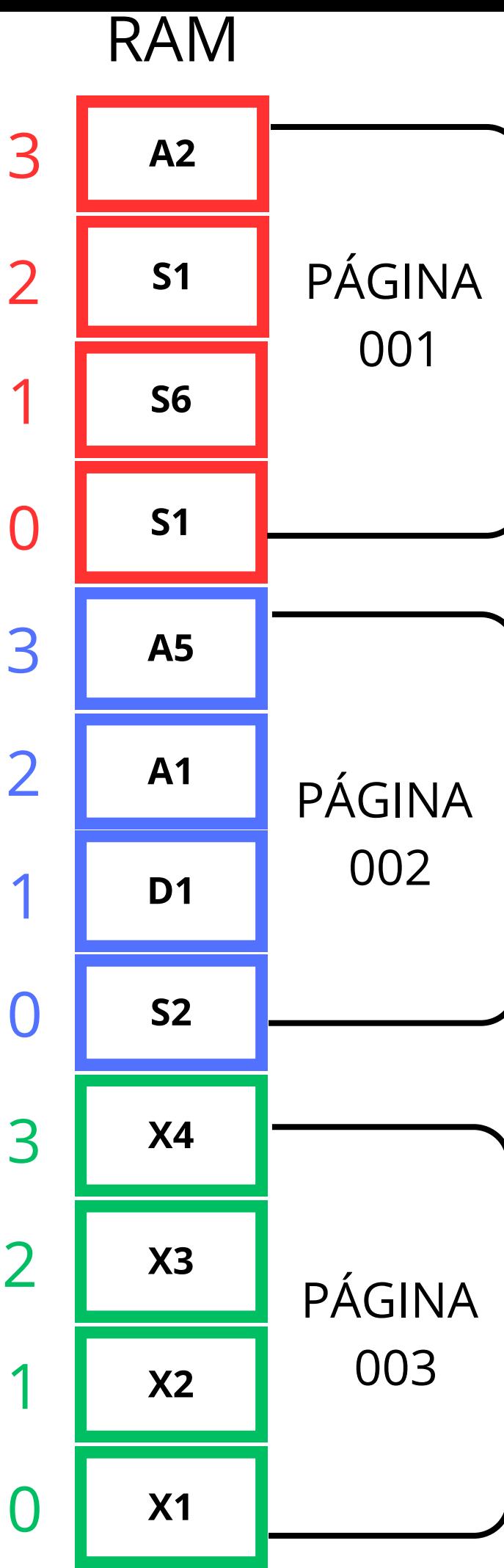
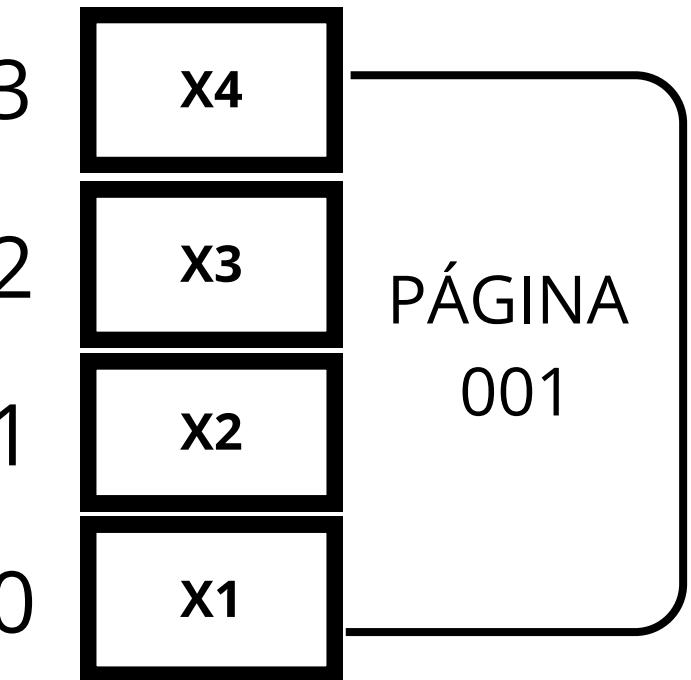
1 - Um programa tenta acessar uma parte da memória.

2 - S.O verifica se a página correspondente está presente na memoria.

    → Se a página estiver presente, ocorre um acesso rápido à RAM.

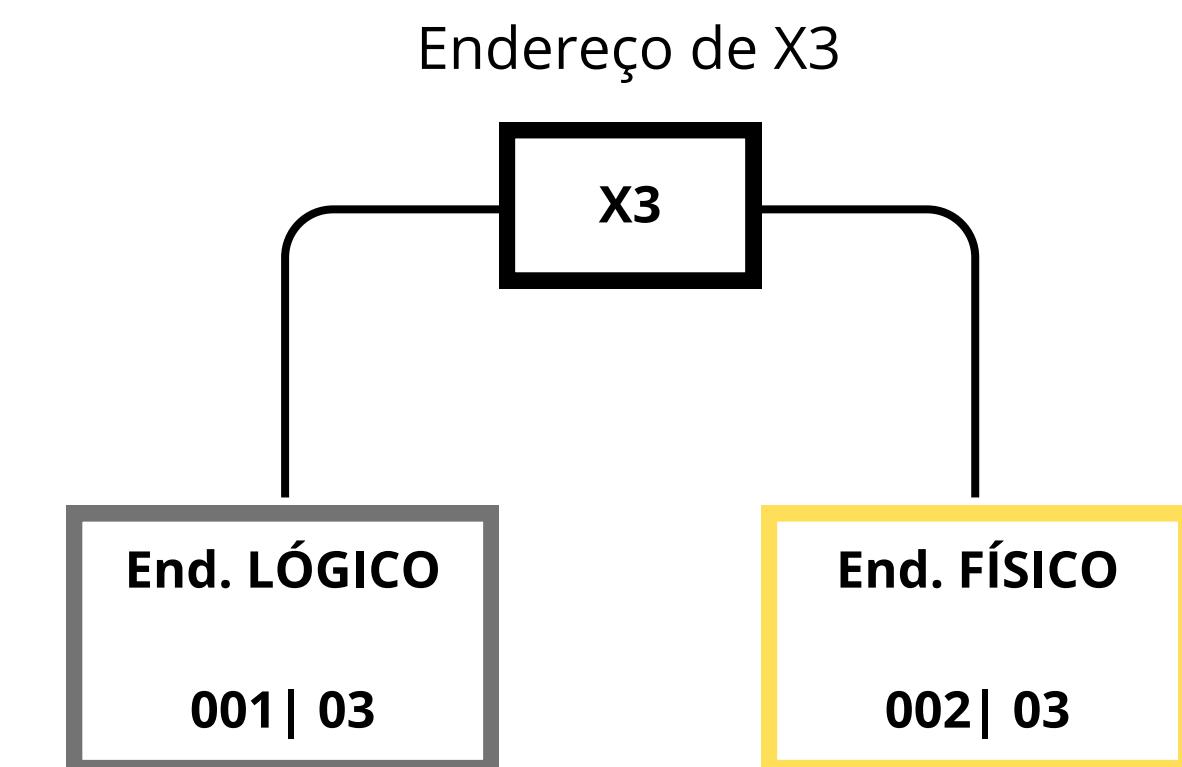
    → Se a página não estiver, ocorre o chamado "page fault" (falha de página).

    → O S.O move a pagina ausente para a memória física substituindo uma página existente se necessário. Esse processo é chamado "troca de páginas".



**Tabela de páginas**

PÁGINA LÓGICA	PÁGINA FÍSICA
001	003
....	....



# Algoritmo NRU

## DEFINIÇÃO

O algoritmo NRU (Not Recently Used) categoriza páginas em **quatro tipos** com base em bits de **referência** e **modificação**

Selecionando **aleatoriamente** para **substituição** dentro da categoria menos prioritária

Embora **simples**, pode ser **menos eficiente** do que algoritmos mais complexos

# Fucionamento

---

1 - Organização de todas as páginas em quatro categorias, após uma page fault.

**Classe 0:** não referenciada  $R = 0$ , não modificada  $M=0$ .

**Classe 1:** não referenciada  $R = 0$ , modificada  $M=1$ .

**Classe 2:** referenciada  $R = 1$ , não modificada  $M=0$ .

**Classe 3:** referenciada  $R = 1$ , modificada  $M = 0$ .

2 - O NRU então remove uma página aleatória da classe mais baixa que não esteja vazia.

# Algoritmo Fifo

## DEFINIÇÃO

Algoritmo First-in First-out Page Replacement.

- São estruturas de dados do tipo FIFO (first-in first-out), onde o primeiro elemento a ser inserido, será o primeiro a ser retirado, ou seja, adiciona-se itens no fim e remove-se do início.

# Fucionamento

---

- Só mantem uma fila de páginas correntes na memória.
- Simples, mas pode ser ineficiente, pois uma página que está em uso constante pode ser retirada
- Processos que se tornam aptos são inseridos no final da fila
- Processo que esta no inicio da fila é o próximo a executar
- Processo executa até que:
  - Libere explicitamente o processador
  - Realize uma chamada de sistema
  - Termine sua execução

# FIFO

---

## Vantagens

- Fácil entendimento e implementação
- Simples de entender e implementar

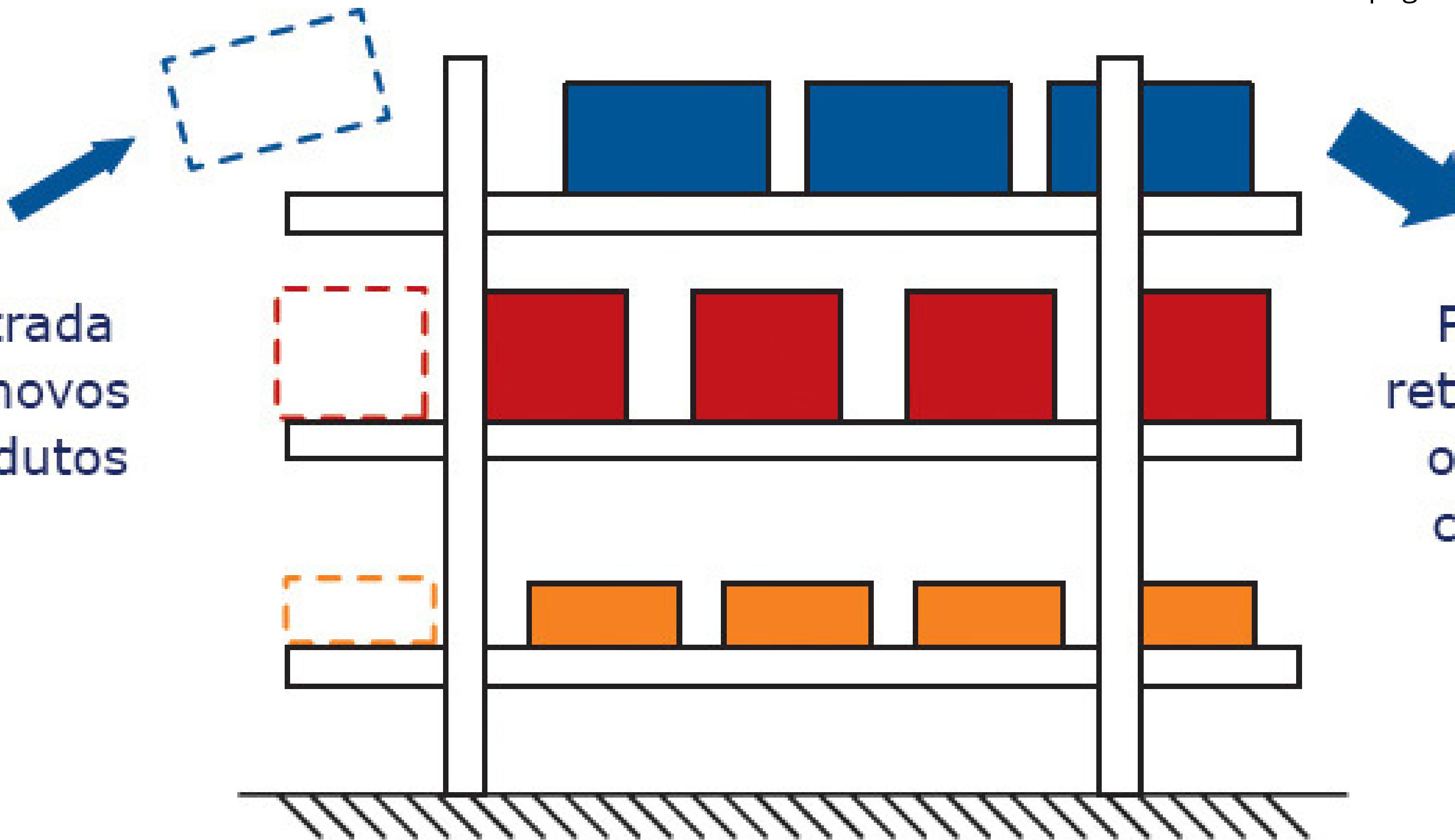
## Desvantagens

- Eficiência do processo é baixa
- Cada quadro precisa ser levado em consideração

Tabela de páginas

Entrada  
de novos  
produtos

Produtos  
retirados por  
ordem de  
chegada.



# Algoritmo do Relógio

## DEFINIÇÃO

Visão melhorar o desempenho do algoritmo RC, diferenciando apenas na implementação da fila

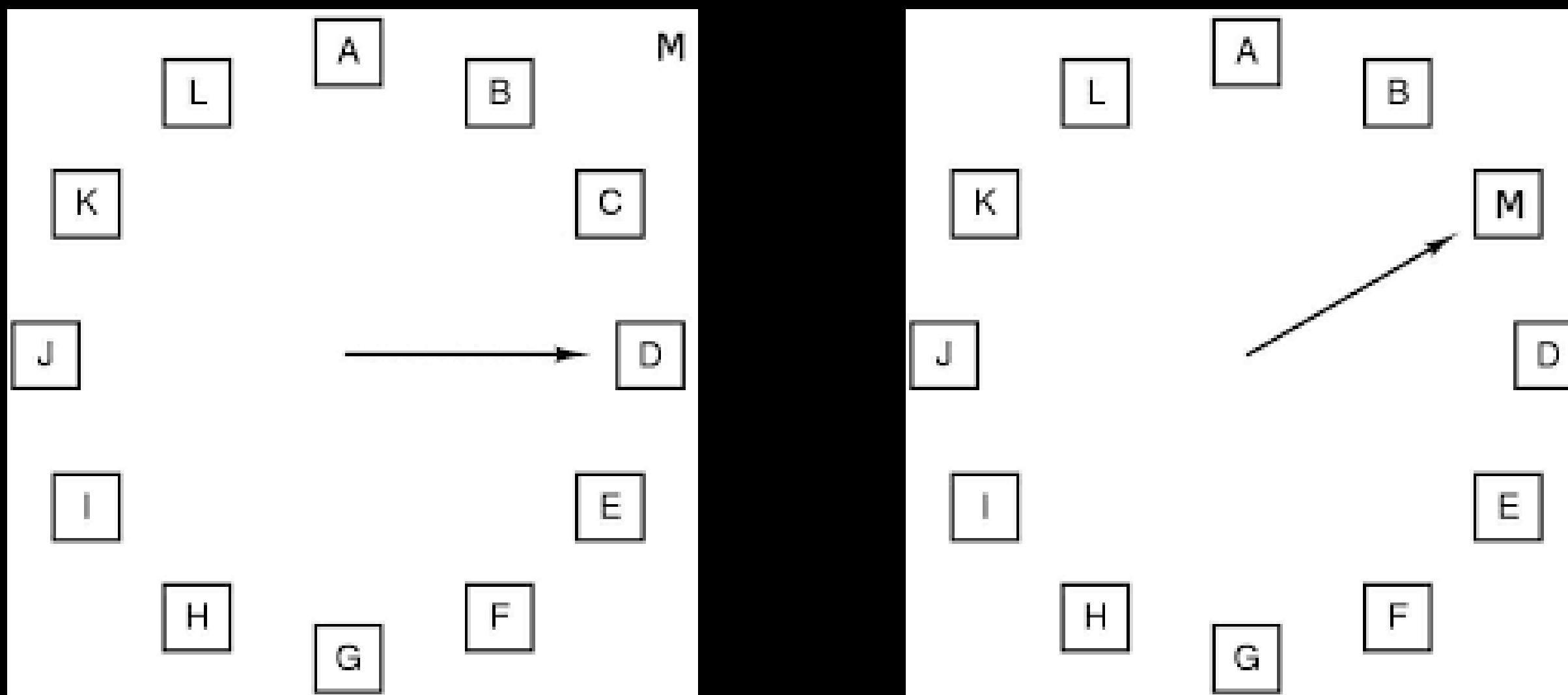
- O ponteiro sempre aponta para a pagina mais antiga
- Se  $R=1$ ,  $R$  é resetado e o ponteiro avança para a proxima pagina até encontrar o  $R=0$

# Fucionamento

---

- Quando ocorre um page fault:
  - Inspeciona-se a cabeça da lista
  - Se  $R = 0$ :
    - Substitui-se a página da cabeça pela nova página
    - Avança-se a cabeça em uma posição
  - Se  $R = 1$ :
    - Avança-se a cabeça em uma posição
    - Repete-se o processo até encontrar página com  $R = 0$
    - Age como no caso anterior ( $R=0$ )

# Fucionamento



# Algoritmo LFU

## DEFINIÇÃO

LFU significa Least Frequently Used . É um algoritmo de cache usado para otimizar a memória do computador.

Nesse algoritmo, o sistema mantém um registro de quantas vezes um bloco é referido na memória. Quando a memória cache está completamente cheia e exige mais espaço, o item com a frequência de referência mais baixa é removido.

Três operações executadas pelo LFU são  
- Set (inserir), Recuperar (consultar),  
Remover (excluir).

# Fucionamento

---

## O algoritmo LFU

- Pode seguir o método FIFO, ou seja, primeiro a entrar, primeiro a sair.
- Min Heap pode ser usado para implementar este algoritmo na complexidade de tempo logarítmico.
- É muito sistemático nos casos em que o padrão de acesso para a memória cache não muda com muita frequência.

# Algoritmo LFU

---

## Vantagens

- Útil para encontrar um item quando os dados se repetem
- No LFU, a página antiga junto com a frequência é verificada para essa página.

## Desvantagens

- Novos itens inseridos no cache serão removidos em breve, pois terão baixa contagem, mas podem ser usados com muita frequência novamente.
- A anomalia de Belady também pode ocorrer

# Fucionamento

3	5	6	6	1	1	4	2	3	4
3	3	3	3	1	1	1			
	5	5	5	5	5	4			
		6	6	6	6	6			

P

**FREQUENCIES AT P :-**

**3 = 0      5 = 0      6 = 2      1 = 2      4 = 1**

# Algoritmo LRU

## DEFINIÇÃO

Algoritmo Least Recently Used Page Replacement:  
é um algoritmo de pilha em que o critério de escolha da página indica que a página excluída será aquela que não é referenciada há mais tempo. Optimal, também é um algoritmo de pilha, mas escolhe para sair a página que levará mais tempo para ser novamente necessária.

# Funcionamento

---

## O algoritmo LRU

- Idade da página: páginas muito antigas talvez sejam pouco usadas.
- Frequência de acessos: páginas muito acessadas possivelmente ainda o serão.
- Data do último acesso: páginas sem acesso a muito tempo não o serão mais.
- Prioridade: processos de alta prioridade, ou de tempo-real, podem precisar de suas páginas de memória rapidamente.
- Conteúdo da página: páginas com código executável exigem menos esforço enquanto páginas de dados que tenham sido alteradas precisam ser salvas.
- Páginas especiais: páginas contendo buffers de operações de E/S causam problemas.

# Algoritmo LRU

---

## Vantagens

- Ao contrário do FIFO, o LRU não sofre da anomalia de Belady.
- Ele fornece menos número de falhas de página do que qualquer outro algoritmo diferente do ideal e, como o algoritmo ideal não pode ser implementado no LRU da vida real, é o algoritmo usado com mais frequência.
- O algoritmo LRU é muito eficiente.

## Desvantagens

- Há mais sobrecarga, pois tem que controlar quais páginas foram referenciadas.
- É difícil de implementar, pois é necessária assistência de hardware.

# Fucionamento

---

	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

	0	1	2	3
0	0	0	1	1
1	0	1	1	1
2	0	0	0	0
3	0	0	0	0

	0	1	2	3
0	0	0	0	1
1	0	0	1	1
2	1	1	0	1
3	0	0	0	0

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	1	1	0	0
3	1	1	1	0

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	1	1	0	1
3	1	1	0	0

## **Reconhecimentos e Direitos Autorais**

**@autor:** MARIA HELENA DE SOUSA COSTA  
LAIS SILVA COSTA  
ARTHUR SALIM DA COSTA

**@ contato:** maria.hsc@discente.ufma.br  
lais.sc@discente.ufma.br  
arthur.salim@discente.ufma.br

**@data última versão:** [10.12.2023]

**@versão:** 1.0

**@Agradecimentos:** Universidade Federal do Maranhão (UFMA), Professor Doutor Thales Levi Azevedo Valente, e colegas de curso.

**@Copyright/License**

Este material é resultado de um trabalho acadêmico para a disciplina **SISTEMAS OPERACIONAIS**, sobre a orientação do professor **Dr. THALES LEVI AZEVEDO VALENTE**, semestre letivo 2023.2, curso Engenharia da Computação, na Universidade Federal do Maranhão (UFMA). Todo o material sob esta licença é software livre: pode ser usado para fins acadêmicos e comerciais sem nenhum custo. Não há papelada, nem royalties, nem restrições de "copyleft" do tipo **GNU**. Ele é licenciado sob os termos da licença MIT reproduzida abaixo e, portanto, é compatível com **GPL** e também se qualifica como software de código aberto. É de domínio público. Os detalhes legais estão abaixo. O espírito desta licença é que você é livre para usar este material para qualquer finalidade, sem nenhum custo. O único requisito é que, se você usá-lo, nos dê crédito.

Copyright © 2023 Educational Material

Este material está licenciado sob a Licença MIT. É permitido o uso, cópia, modificação, e distribuição deste material para qualquer fim, desde que acompanhado deste aviso de direitos autorais.

O MATERIAL É FORNECIDO "COMO ESTÁ", SEM GARANTIA DE QUALQUER TIPO, EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS DE COMERCIALIZAÇÃO, ADEQUAÇÃO A UM DETERMINADO FIM E NÃO VIOLAÇÃO. EM HIPÓTESE ALGUMA OS AUTORES OU DETENTORES DE DIREITOS AUTORAIS SERÃO RESPONSÁVEIS POR QUALQUER RECLAMAÇÃO, DANOS OU OUTRA RESPONSABILIDADE, SEJA EM UMA AÇÃO DE CONTRATO, ATO ILÍCITO OU DE OUTRA FORMA, DECORRENTE DE, OU EM CONEXÃO COM O MATERIAL OU O USO OU OUTRAS NEGOCIAÇÕES NO MATERIAL.

Para mais informações sobre a Licença MIT: <https://opensource.org/licenses/MIT>.

# Referências

---

- BERENGER, Fransis. MAIA, Luís. Arquitetura de Sistemas Operacionais. 5<sup>a</sup> edição. Editora LTC, Rio de Janeiro, 2013.
- TANENBAUM, A. S. Sistemas operacionais modernos. Rio De Janeiro (Rj): Prentice-Hall Do Brasil, 2010.
- SOUZA. A, RAIBIDA. T. **FIFO - UMA ABORDAGEM SIMPLES.** Disponivel em: <https://deinfo.uepg.br/~alunoso/2016/FIFO/>. Acesso em: 11 de nov, 2023.
- ROMAN, N. MORANDINI, M, UEMAYA, J. Disponivel em: <http://wiki.icmc.usp.br/images/d/dc/Aula12.pdf>. Acesso em 10 de nov. 2023