

SISTEMAS OPERACIONAIS

AULA 07: GERENCIAMENTO DE MEMÓRIA

26 de junho de 2025

Prof. Me. José Paulo Lima

IFPE Garanhuns



**INSTITUTO
FEDERAL**
Pernambuco

SUMÁRIO



Memórias do Computador

- Hierarquia das memórias

- Trabalho do Sistema Operacional

- Desafios

- Abstração de memória

 - Nenhuma abstração

 - Espaço de endereçamento

 - Troca de memória (*swapping*)

Gerenciamento de Memória

- Mapa de Bits

- Listas Encadeadas

- Algoritmos de Gerenciamento

 - First Fit*

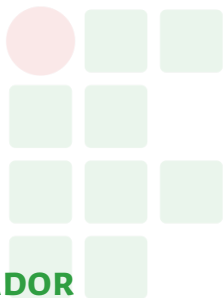
 - Next Fit*

 - Best Fit*

 - Worst Fit*

 - Quick Fit*

Referências



MEMÓRIAS DO COMPUTADOR

**INSTITUTO
FEDERAL**
Pernambuco

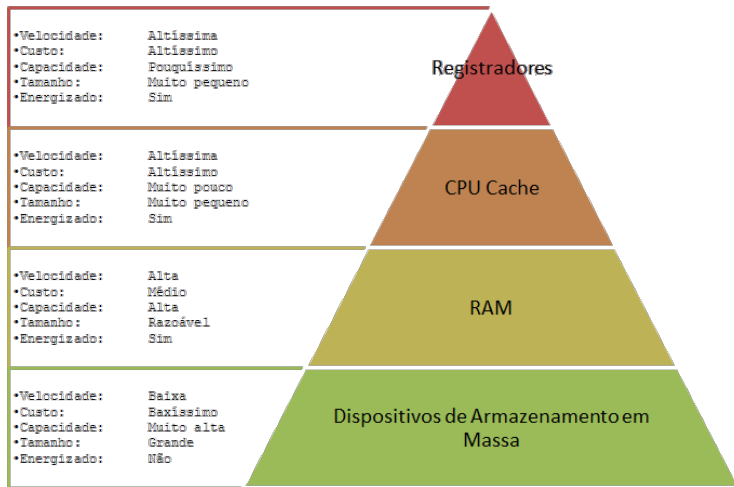
MEMÓRIAS DO COMPUTADOR



- ▶ Memória RAM é um recurso importante mas deve ser gerenciado com cuidado;
 - ▶ Recurso caro e escasso;
 - ▶ Programas precisam ser carregados na memória principal para ser executado;
 - ▶ Carrega o programa da memória secundária para memória principal.
 - ▶ Quanto mais processos alocados na memória maior será o desempenho da CPU;
 - ▶ Por motivo de preço, buscou-se utilizar uma hierarquia de memória.

MEMÓRIAS DO COMPUTADOR

HIERARQUIA DAS MEMÓRIAS



GERENCIAMENTO DE MEMÓRIA

TRABALHO DO SISTEMA OPERACIONAL



- ▶ A função do sistema operacional é abstrair essa hierarquia em modelo útil;
 - ▶ A parte do sistema operacional que gerencia a hierarquia de memória é denominado gerenciador de memória;
 - ▶ A função do gerenciador de memória é gerenciar as memórias de forma eficiente;
 - ▶ Manter o controle das partes da memória que estão livres ou alocadas aos processos e liberá-las quando terminam.
- ▶ Multiprogramação corresponde a manter de vários processos em memória;
 - ▶ A memória precisa ser distribuída de forma eficiente para permitir o máximo de processos possível.
- ▶ Existe diversos algoritmos para gerenciamento de memória.

GERENCIAMENTO DE MEMÓRIA

DESAFIOS



- ▶ Programas tornam-se maior muito mais rápido do que as memórias;
- ▶ Todo programador deseja dispor de uma memória infinitivamente grande, rápida e não volátil;
- ▶ A memória pode ser gerenciada de formas diferentes:
 - ▶ Um processo ocupa toda a memória;
 - ▶ Ou cada processo é alocado em uma parte da memória.
- ▶ Processos não devem poder referenciar posições de memória em outros processos sem permissão.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: NENHUMA ABSTRAÇÃO



- ▶ A abstração de memória mais simples é não ter abstração alguma;
 - ▶ Cada programa simplesmente considerava a memória física;
 - ▶ Não era permitido executar dois programas na memória simultaneamente;
 - ▶ Exemplo:
 - ▶ Se um programa escrevesse um valor na memória na posição 4000 que outro programa estivesse usando, apagaria o valor anterior.
- ▶ Até 1980 os computadores não tinham abstração de memória. Cada programa apenas via a memória física, mesmo assim, várias opções eram possíveis:

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: NENHUMA ABSTRAÇÃO



Usado em computadores de grande porte, minicomputadores e raramente foi utilizado depois.

Usado em alguns computadores portáteis e sistemas embarcados.

Usado nos primeiros computadores pessoais.

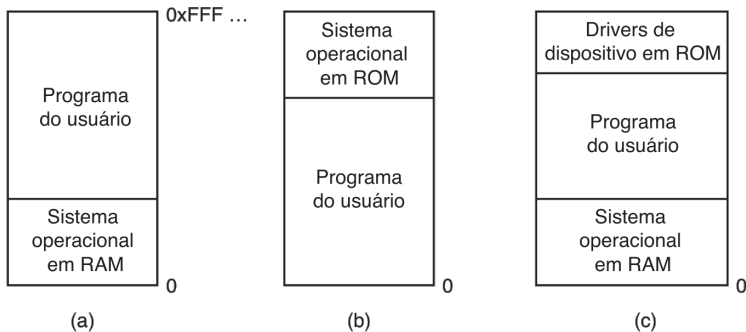


Figura extraída de Tanenbaum e Bos (2016, p. 126).

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: NENHUMA ABSTRAÇÃO



- ▶ Uma maneira de se ter algum paralelismo em um sistema sem abstração de memória é programá-lo com múltiplos *threads*;
 - ▶ Para executar o sistema operacional deve salvar o conteúdo completo da memória em um arquivo no disco e, em seguida, carregar e executar o próximo programa;
 - ▶ Processo custoso que faz a CPU ficar ociosa até o novo programa seja totalmente carregado para ser executado.
- ▶ A exposição da memória física apresenta várias desvantagens:
 - ▶ Se os programas usuários podem endereçar a memória, então, podem danificar o sistema e paralisá-lo;
 - ▶ É difícil executar múltiplos programas simultaneamente, principalmente se tiver apenas uma CPU.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: ESPAÇO DE ENDEREÇAMENTO



- ▶ Para que múltiplas aplicações estejam na memória simultaneamente sem interferência mutua, dois problemas devem ser resolvidos:
 - ▶ Proteção;
 - ▶ Realocação.
- ▶ Solução primitiva para proteção usada no IBM 360:
 - ▶ Rotular blocos de memória com uma chave de proteção e comparar a chave do processo em execução com a de cada palavra da memória.
 - ▶ Uma solução lenta e complicada.
 - ▶ Um solução melhor é inventar uma abstração para a memória:
 - ▶ O espaço de endereçamento cria um tipo de memória abstrata para abrigá-los.

GERENCIAMENTO DE MEMÓRIA

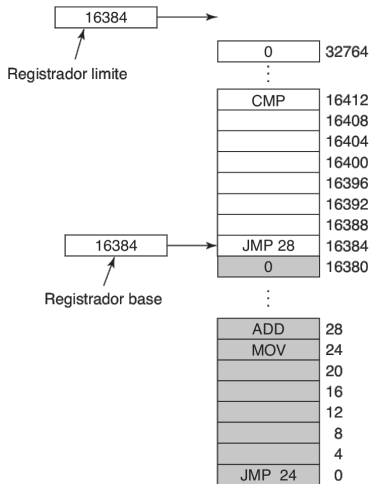
ABSTRAÇÃO DE MEMÓRIA: ESPAÇO DE ENDEREÇAMENTO



- ▶ Espaço de endereçamento:
 - ▶ Conjunto de endereços que um processo pode usar para endereçar memória;
 - ▶ Geralmente, processos possuem espaço de endereçamento próprios.
 - ▶ Exceto quando se deseja compartilhar a memória.
- ▶ Exemplos de “espaços de endereçamento”:
 - ▶ Telefones;
 - ▶ Placas de carro;
 - ▶ Endereços IP.
- ▶ Em memória de computadores:
 - ▶ O endereço 28 para um processo não é o mesmo endereço 28 para outro processo na memória física.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: ESPAÇO DE ENDEREÇAMENTO



- ▶ **Registrador-base**
 - ▶ Guarda o endereço físico inicial do programa.
- ▶ **Registrador-limite**
 - ▶ Guarda comprimento do programa.
- ▶ **Desvantagem da realocação usando registradores base e limite:**
 - ▶ Necessidade de executar uma adição e uma comparação em cada referência à memória.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: ESPAÇO DE ENDEREÇAMENTO



1. Na figura anterior, o registrador-base e o registrador-limite contêm o mesmo valor, 16.384. Isso é apenas um acidente ou eles são sempre iguais? Se for apenas um acidente, por que eles são iguais nesse exemplo?
 - ▶ É um acidente, o registrador-base é 16.384 porque o programa passou a ser carregado no endereço 16.384. Poderia ter sido carregado em qualquer lugar;
 - ▶ O registrador-limite é 16.384 porque o programa contém 16.384 bytes, poderia ter tido qualquer comprimento;
 - ▶ É pura coincidência que o endereço inicial corresponde exatamente ao comprimento do programa.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA

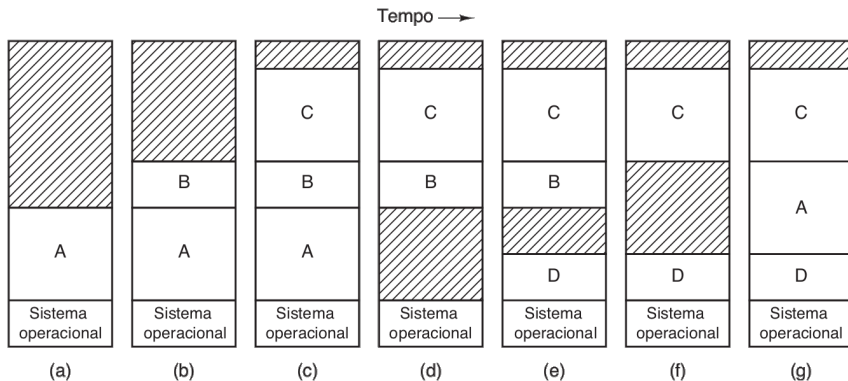


- ▶ Se a memória física for grande o suficiente para armazenar todos os processos, os esquemas anteriores bastarão;
 - ▶ Na prática a quantidade total de RAM necessária para todos os processos é normalmente muito maior do que pode comportar;
 - ▶ Principalmente com a grande quantidade de processos que são iniciados juntos com os sistemas operacionais;
- ▶ Dois métodos gerais tem sido desenvolvidos para lidar com a sobrecarga de memória:
 - ▶ *Swapping*:
 - ▶ Consiste em trazer, em sua totalidade, cada processo para a memória e executá-lo durante um certo tempo e, então, devolvê-lo ao disco.
 - ▶ Memória virtual:
 - ▶ Permite que programas possam ser executados mesmo que estejam apenas parcialmente carregados na memória principal.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA

Mudanças na alocação de memória à medida que processos entram nela e saem dela. As regiões sombreadas são regiões não utilizadas da memória.



GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA

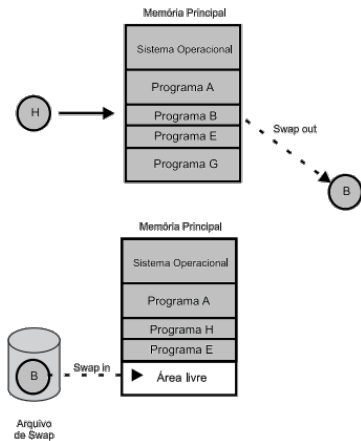


Figura extraída de Tanenbaum e Bos (2016, p. 168).

Definição:

“O *swapping* é uma técnica aplicada à gerência de memória principal para a memória secundária (*swap out*), geralmente disco. Posteriormente, o processo é carregado de volta da memória secundária para a memória principal (*swap in*) e pode continuar sua execução como se nada tivesse ocorrido.” (MACHADO; MAIA, 2017, p. 167).

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA



- ▶ Quando as trocas de processos deixam muitos espaços vazios na memória é possível combiná-los todos em um espaço contíguo de memória;
 - ▶ Movendo os processos o máximo possível para os endereços mais baixos;
 - ▶ Essa técnica é denominada **compactação de memória**.
 - ▶ Ela geralmente não é usada em virtude do tempo de processamento necessário.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA



2. Quanto tempo (em segundos) uma máquina com 1GB de memória e que possa copiar (tempo total de leitura e escrita) 4 bytes em 20 ns gastaria para compactar toda a memória?

► Dados:

► 1GB = 2^{30} bytes = 1.073.741.824 bytes.

► 1 ns = 10^{-9} s.

► Resposta:

$$\begin{array}{rcl} 4 \text{ bytes} & - & 20 \text{ ns} \\ 1 \text{ GB} & - & ? \end{array}$$

$$\begin{array}{rcl} 4 \text{ bytes} & - & 20 \times 10^{-9} \text{ s} \\ 2^{30} \text{ bytes} & - & x \end{array}$$

$$\begin{aligned} x &= \frac{2^{30} \times 20 \times 10^{-9}}{4} \\ x &= 2^{30} \times 5 \times 10^{-9} \\ x &= 1.073.741.824 \times 5 \times 10^{-9} \\ x &= 1,073741824 \times 5 \\ x &\cong 5,37 \text{ segundos} \end{aligned}$$

GERENCIAMENTO DE MEMÓRIA

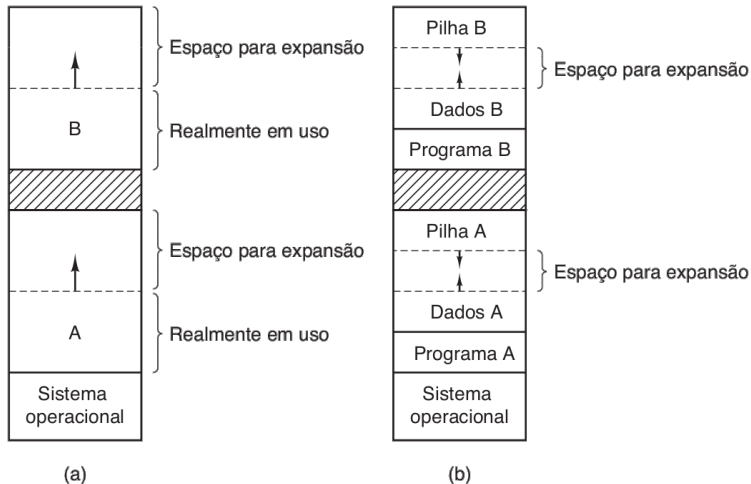
ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA

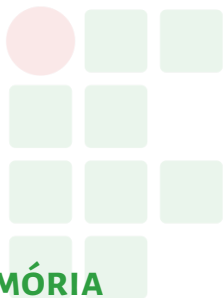


- ▶ Um ponto importante é a quantidade de memória alocada;
- ▶ Se processos são criados com tamanhos fixo, a alocação é simples;
- ▶ Se a área dos processos puderem crescer, problemas poderão ocorrer sempre que um processo tentar crescer.
 - ▶ Se um espaço livre adjacente ao processo, ele poderá ser alocado, o processo cresce;
 - ▶ Por outro lado, se estiver adjacente a outro processo, o processo que necessita crescer poderá ser movido para uma área de memória grande;
 - ▶ Se não tiver espaço suficiente o processo deverá ser suspenso até que algum espaço seja liberado.

GERENCIAMENTO DE MEMÓRIA

ABSTRAÇÃO DE MEMÓRIA: TROCA DE MEMÓRIA





GERENCIAMENTO DE MEMÓRIA

**INSTITUTO
FEDERAL**
Pernambuco

GERENCIAMENTO DE MEMÓRIA LIVRE

MAPA DE BITS



- ▶ Quando há atribuição de memória dinamicamente, o sistema operacional deve gerenciá-la através de:
 - ▶ Mapa de bits;
 - ▶ A memória é dividida entre unidades de alocação tão pequenas quanto palavras ou grandes com vários Kilobytes;
 - Cada alocação corresponde a um bit no mapa de bits (0 para a unidade livre e 1 se tiver ocupada);
 - Mesmo com uma unidade de alocação tão pequena de memória será necessário 1 bit no mapa de bits;
 - Se a unidade de alocação for grande precisará de menos bits mas a memória será desperdiçada;
 - ▶ Problema com essa técnica: quando se decide carregar na memória um processo tamanho de K unidades o gerenciador de memória precisa encontrar um espaço livre de K bits consecutivos em '0'.
- ▶ Lista encadeadas.
 - ▶ Mantém uma lista encadeada de segmentos;
 - Um segmento é tanto uma área de memória alocada a um processo como uma área de memória livre.

GERENCIAMENTO DE MEMÓRIA LIVRE

MAPA DE BITS

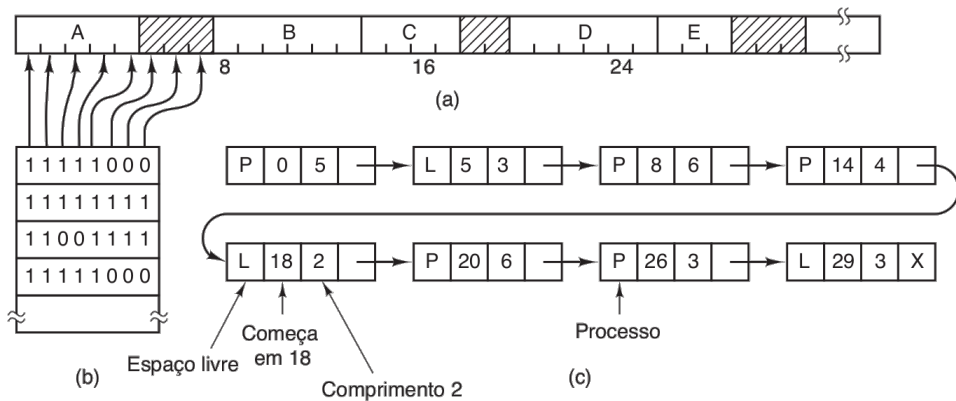


Figura extraída de Tanenbaum e Bos (2016, p. 132).

GERENCIAMENTO DE MEMÓRIA LIVRE

LISTAS ENCADEADAS

Um processo que termina sua execução geralmente ter dois vizinhos na lista encadeada (exceto quando estiver no início ou no fim)

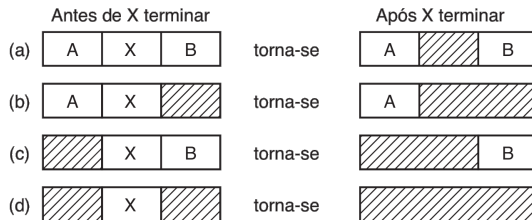


Figura extraída de Tanenbaum e Bos (2016, p. 132).

ALGORITMOS DE GERENCIAMENTO



Quando os segmentos de memória alocados a processos e segmentos de memória livre são mantidos, é possível utilizar diversos algoritmos para alocar um novo processo ou carregar para a memória:

- ▶ *First Fit;*
- ▶ *Next Fit;*
- ▶ *Best Fit;*
- ▶ *Worst Fit;*
- ▶ *Quick Fit.*

ALGORITMOS DE GERENCIAMENTO

FIRST FIT



Definição:

Na estratégia *first-fit*, a primeira partição livre de tamanho suficiente para carregar o programa é escolhida. (MACHADO; MAIA, 2017, p. 167).

- ▶ Algoritmo mais simples;
 - ▶ É o mais rápido porque procura o menos possível;
 - ▶ O espaço livre é quebrado em memória livre.

ALGORITMOS DE GERENCIAMENTO

NEXT FIT



- ▶ Próximo encaixe;
 - ▶ Funciona da mesma maneira que o *First Fit*, exceto pelo fato de sempre memorizar a posição em que encontra um segmento de memória disponível;
 - ▶ Quando for chamado ele inicia sua busca a partir desse ponto.

ALGORITMOS DE GERENCIAMENTO

BEST FIT



Definição:

“Na estratégia *best-fit*, a melhor partição é escolhida, ou seja, aquela em que o programa deixa o menor espaço sem utilização.” (MACHADO; MAIA, 2017, p. 165).

► Problemas:

- Gera muitos segmentos de memória de tamanho pequeno;
 - Para evitar o problema de alocar um segmento de memória disponível de tamanho quase exato ao requisitado pelo processo e gerar minúsculos segmentos de memória disponível;
 - Seria possível pensar em um algoritmo que alocasse no maior segmento para que o restante do segmento disponível pudesse alocar outro processo.
- É mais lento que o algoritmos *First Fit*, pois precisa pesquisara lista inteira cada vez que for chamado.

ALGORITMOS DE GERENCIAMENTO

WORST FIT



Definição:

“Na estratégia *worst-fit*, a pior partição é escolhida, ou seja, aquela em que o programa deixa o maior espaço sem utilização.” (MACHADO; MAIA, 2017, p. 165).

- ▶ Pior encaixe;
 - ▶ A ideia é que o segmento de memória disponível que sobrasse fosse suficiente para alocar outro processo.
- ▶ Simulações mostram que o algoritmo não é uma ideia muito boa.

ALGORITMOS DE GERENCIAMENTO

QUICK FIT



- ▶ Encaixe mais rápido;
 - ▶ O algoritmo mantém listas separadas para alguns dos tamanhos de segmentos de memória disponíveis, em geral mais solicitados.
- ▶ Com o *Quick Fit*, buscar um segmento de memória disponível de um determinado tamanho é extremamente rápido, mas apresenta a mesma desvantagem de todos os esquemas:
 - ▶ É dispendioso descobrir quais são os segmentos de memória disponíveis.

ALGORITMOS DE GERENCIAMENTO

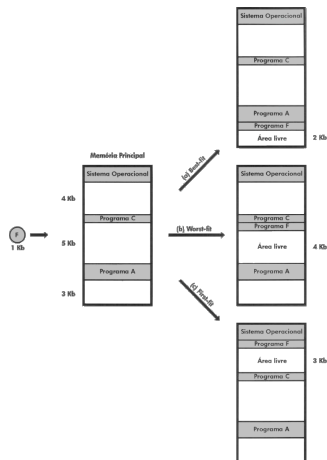


Figura extraída de Machado e Maia (2017, p. 166).

ALGORITMOS DE GERENCIAMENTO

EXERCÍCIO



3. Considere um sistema de troca de processos entre a memória e o disco no qual a memória é constituída dos seguintes tamanhos de lacunas em ordem de memória:

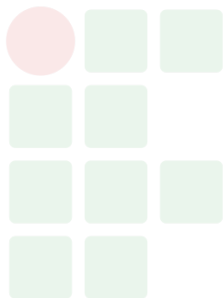
- ▶ 10KB;
- ▶ 4KB;
- ▶ 20KB;
- ▶ 18KB;
- ▶ 7KB;
- ▶ 9KB;
- ▶ 12KB;
- ▶ 15KB.

Qual lacuna é tomada pelas solicitações sucessivas do segmento de tamanhos:

- ▶ 12KB
 - ▶ 10KB
 - ▶ 9 KB
- a) Para o *First Fit*?
- b) Para o *Best Fit*?
- c) Para o *Worst Fit*?
- d) Para o *Next Fit*?

Solicitação	<i>First Fit</i>	<i>Best Fit</i>	<i>Worst Fit</i>	<i>Next Fit</i>
12KB	20KB	12KB	20KB	20KB
10KB	10KB	10KB	18KB	18KB
9KB	18KB	9KB	15KB	9KB



REFERÊNCIAS

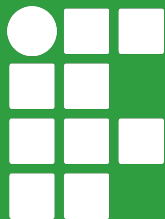


INSTITUTO
FEDERAL
Pernambuco

REFERÊNCIAS I



-  MACHADO, Francis Berenger; MAIA, Luiz Paulo. **Arquitetura de Sistemas Operacionais**. 5. ed. Rio de Janeiro: LTC, 2017. ISBN 978-85-216-2210-9.
-  TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016.



INSTITUTO FEDERAL

Pernambuco