Universidade Federal de Minas Gerais

Vitor Ribeiro dos Santos - 2017023676

Igor Oliveira Valente da Silveira - 2017105575

TRABALHO PRÁTICO 2

1. Introdução

O objetivo desse trabalho é a implementação dos algoritmos de substituição simulando uma memória virtual. A ideia desse simulador é a comparação entre os diferentes tipos de algoritmos a fim de identificar os resultados e obter respostas significativas entre esses algoritmos e seu funcionamento em uma memória virtual.

2. Estruturas e Organização

Para implementação do projeto utilizamos um conjunto de estruturas para armazenar e facilitar os dados coletados dos arquivos de log recebidos pelo programa.

Os dados contidos e cada estrutura, são, respectivamente:

- Pagina: O índice de quadros ligados à página, número da página, ultimo endereço que foi acessado, se a mesma está suja ou não.
- Tabela: Na tabela temos o número de entradas e a estrutura de páginas.
- Quadro: Armazenamos o último acesso, o momento de carregamento para a memória e informação se o quadro está sendo utilizado ou não.

O próximos elementos são utilizados exclusivamente para a implementação do FIFO

- StructItem: Armazena a página e um ponteiro para o próximo item
- Lista: Guarda o tamanho e ponteiros para início e o fim da lista.

Também vinculado a essas estruturas temos a função inserir e remover para manipulação da lista.

Divisão dos arquivos

Optamos por dividir os algoritmos de substituição em diferentes arquivos a fim de obter um código mais claro e de fácil entendimento. No código será encontrado os arquivos:

- fifo.c/fifo.h: Contem a função de execução do algoritmo FIFO e a função de exibir os dados da tabela.
- segundachance.c/segundachance.h: Contém a função de execução do algoritmo Segunda Chance a função de exibir os dados da tabela.
- lru.c/lru.h: Contém a função de execução do algoritmo LRU e a função de exibir os dados da tabela.
- media.c/media.h Contém a função de execução do algoritmo Media e a função de exibir os dados da tabela.

- estruturas.c/estruturas.h: Contém as estruturas citadas anteriormente.
- comum.c/comum.h: Contém os métodos comuns que os algoritmos possuem.
- main.c: Funcionamento geral do código

3. Implementação

O arquivo principal main.c será responsável por:

- Recuperar as informações de entrada especificada no trabalho: Algoritmo de substituição, arquivo, tamanho de cada página, e tamanho total da memória.
- Fazer a validação dos dados recuperados.
- Inicializar as estruturas para executar os algoritmos de substituição de acordo com o que foi solicitado
- Resgatar endereço e operação lida em cada arquivo
- Cria um contador que salva número máximo de bits que podem ser usados para identificar as páginas (algoritmo foi retirado da especiação do trabalho).
- Faz o processo de leitura do arquivo fornecido e chama a função de algoritmo de substituição de acordo com o que foi repassado
- Printa a saída dos dados gerais de acordo com a específico fornecida no trabalho.

Agora, revisando o funcionamento de cada uma dos algoritmos de substituição:

3.1 FIFO

Basicamente enquanto o algoritmo lê as entradas do endereço e da operação do arquivo e caso ele tenha selecionado o algoritmo FIFO, ele verifica se a página está na fila e se a página criada está na tabela de página ele atualiza os dados da tabela da página acessada como por exemplo o último endereço acessado e se a página está suja ou não.

Caso a página não esteja na fila ele verifica se todos os quadros estão lotados, caso esteja lotado ele segue o sugerido pelo algoritmo removendo o primeiro endereço acessado da lista. e se caso a página não tenha sido criada ele insere a página na fila com seus respectivos dados coletados.

Complexidade apresentada: O(n)

3.2 LRU

Com um processo semelhante as etapas do algoritmo anterior ele inicia verificando se a página está na tabela, logo depois percorre as páginas inseridas na tabela para atualizar o último endereço, se a página está suja e o último acesso daquela página.

Logo depois é verificado se a página está na tabela ou não e caso não esteja analisamos o quadro para verificar se está cheio ou não. Caso esteja lotado ele segue o ajuste seguido pelo algoritmo removendo o último endereço acessado.

Por fim ele registra os dados coletados do arquivo enviado.

Complexidade apresentada O(n2)

3.3 Segunda Chance

Segue com um processo bem semelhante ao funcionamento do algoritmo LRU, porém seguindo os ajustes do próprio algoritmo criamos uma variável de booleana para utilizar como referência dos bits.

Complexidade apresentada O(n2)

3.4 Média

Segue com um processo bem semelhante ao funcionamento do algoritmo LRU, porém para escolher um novo índice a ser selecionado utilizamos a média do total de página que existem e removemos o valor médio desse resultado.

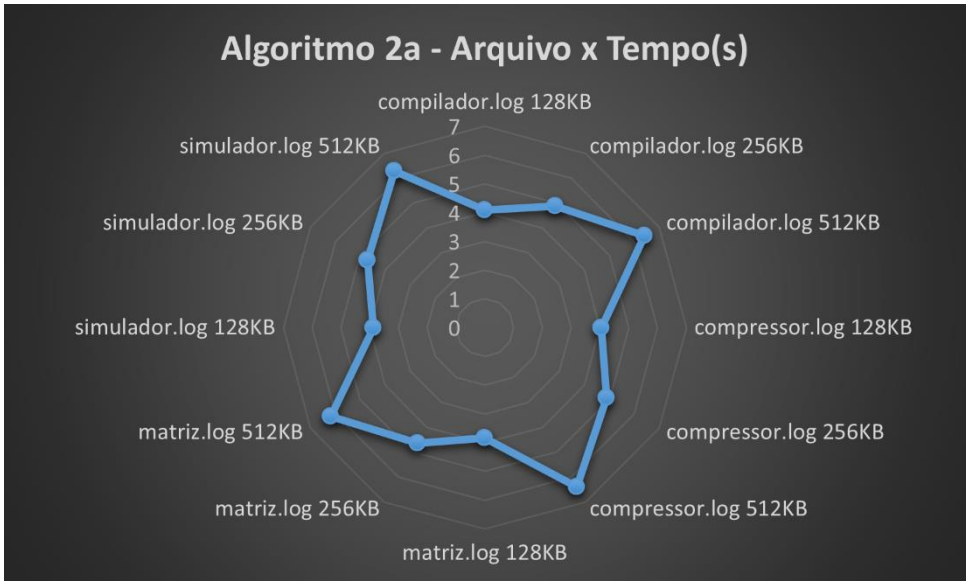
Complexidade apresentada O(n)

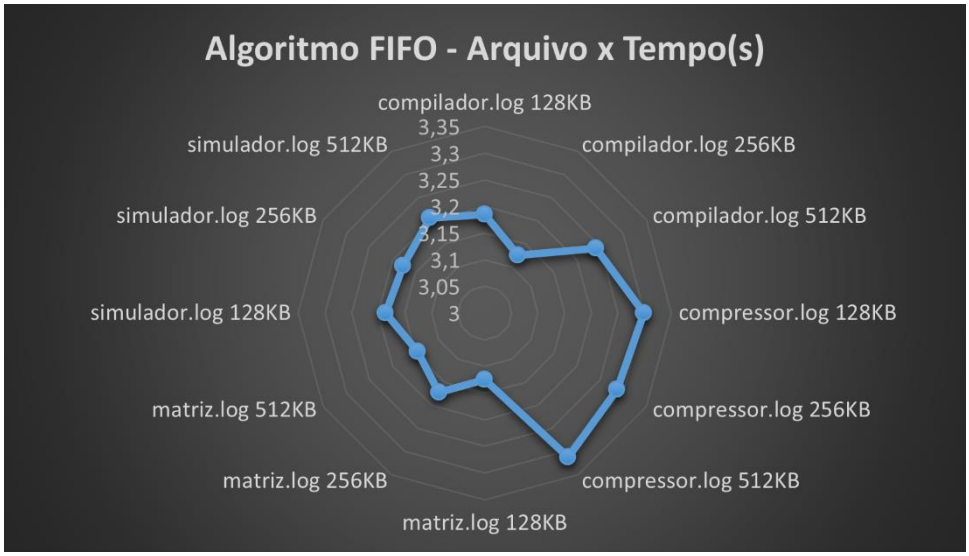
3.5 Decisões de Projeto

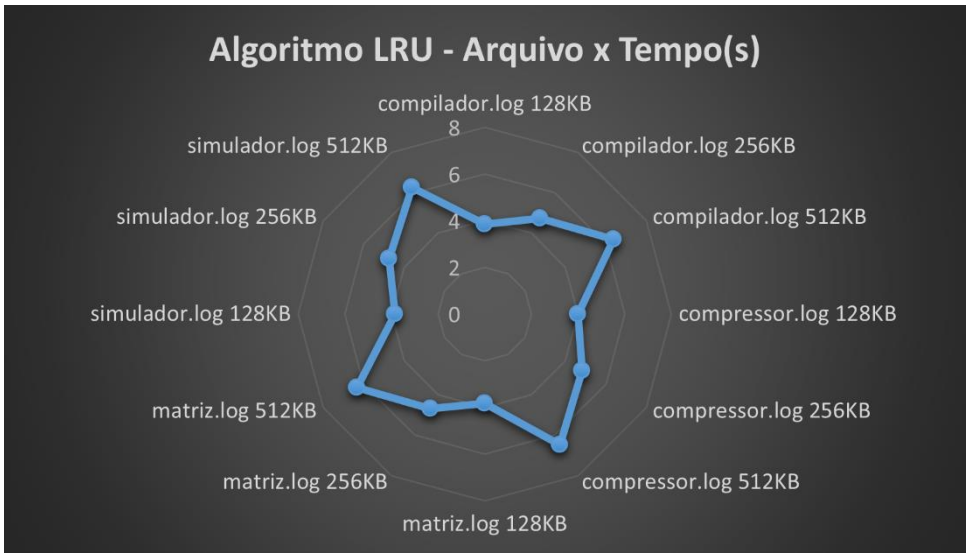
Como decisões do projeto nosso interpretador não escreve em páginas sujas quando não existe a linha de operação de escrita por exemplo. E no caso da tabela optamos por colocar os seguintes campos:

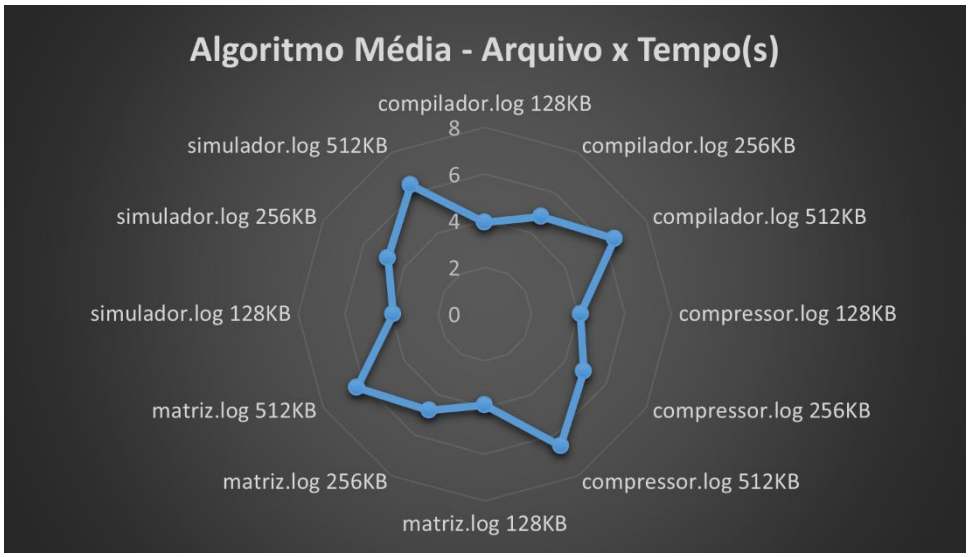
- Endereço: Último endereço acessado
- Número da página lida
- Bit de controle para verificar se a página está suja (SIM) ou (NÃO)

Análise dos resultados








Algoritmo Média - Arquivo x Tempo(s)
compilador.log 128KB
compilador.log 256KB
compilador.log 512KB
compressor.log 128KB
compressor.log 256KB
compressor.log 512KB
matriz.log 128KB
matriz.log 256KB
matriz.log 512KB
simulador.log 128KB
simulador.log 256KB
simulador.log 512KB
8
6
4
2
0