UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
SISTEMAS DE INFORMAÇÃO

CARLOS EDUARDO RODRIGUES DE SOUSA - 11295758
GABRIEL JANSEN MARQUES - 12542741
KAYO RICARDO LIMA DA SILVA - 12543029
LEONARDO BOEING PINHEIRO DE SOUZA - 12542605

SISTEMAS OPERACIONAIS - EXERCÍCIO-PROGRAMA 1
RELATÓRIO

São Paulo
2022

INTRODUÇÃO

Este Exercício-Programa implementa um escalonador de processos para Time Sharing em uma máquina de processador único, utilizando um sistema simples de multiplicação com um algoritmo Round Robin codificado em Java.

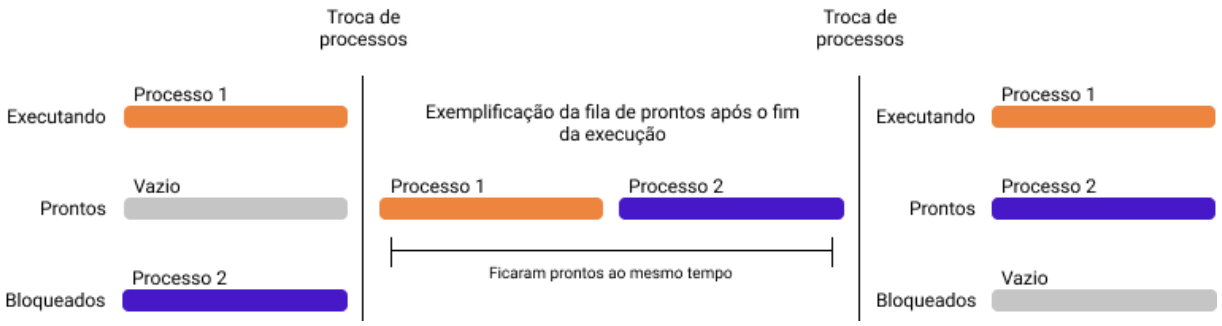
COMPILAÇÃO E EXECUÇÃO DO PROGRAMA

Esta entrega possui compactados todos os arquivos .java necessários para geração dos resultados, para executá-los descompacte em um diretório e execute neste o comando “javac *.java”, que compilará os arquivos e gerará os arquivos .class.

Para executar o programa execute “java Main”. No terminal será solicitado um input do usuário para a quantidade de processos desejados para execução. Para testes foi utilizado o valor de 10 processos localizados na pasta “Programas”.

Durante a implementação houve a seguinte situação: quando a lista de prontos está vazia e o processo-1 em execução termina sua quanta, acontece uma interrupção de este mesmo processo fica pronto novamente. Neste mesmo momento um processo-2 que está bloqueado fica pronto também , isso obriga o escalonador a decidir qual dos dois processos será executado. Para se aproximar o máximo possível de sistemas reais, consideramos a troca de contexto mais custosa do que manter a execução do processo-1. Segue a seguir a esquematização dessa situação na Figura 1.

Figura 1: Decisão de projeto em caso de fila de prontos vazia.



RESULTADOS OBTIDOS

Os programas de teste são compostos cada um por no máximo 21 comandos (incluindo o comando de SAÍDA), para este exercício foram experimentados quanta diferentes de 1 a 21 , com 10 processos. Quanta maiores que 21 não fariam sentido, afinal o número máximo de instruções para um programa é 21.

Para realizar a análise dos resultados utilizamos duas medidas estatísticas: a média de trocas de processo, por processo e a média de instruções executadas por quantum. A seguir são apresentados os resultados em formato de tabela com os quanta e as respectivas médias e, em seguida, os gŕaficos gerados com base nos resultados da Tabela 1.

Tabela 1: Média de trocas de processo e média de instruções por quantum.

Quantum	Média de trocas	Média de instruções
1	13,60	1,00
2	7,70	1,76
3	5,80	2,36
4	5,10	2,62
5	4,40	3,09
6	4,10	3,31
7	3,80	3,56
8	3,80	3,56
9	3,70	3,66
10	3,70	3,66
11	3,50	3,86
12	3,50	3,86
13	3,50	3,86
14	3,40	3,97
15	3,40	3,97
16	3,40	3,97

17	3,40	3,97
18	3,40	3,97
19	3,40	3,97
20	3,40	3,97
21	3,30	4,09

Gráfico 1: Média de trocas de processo.

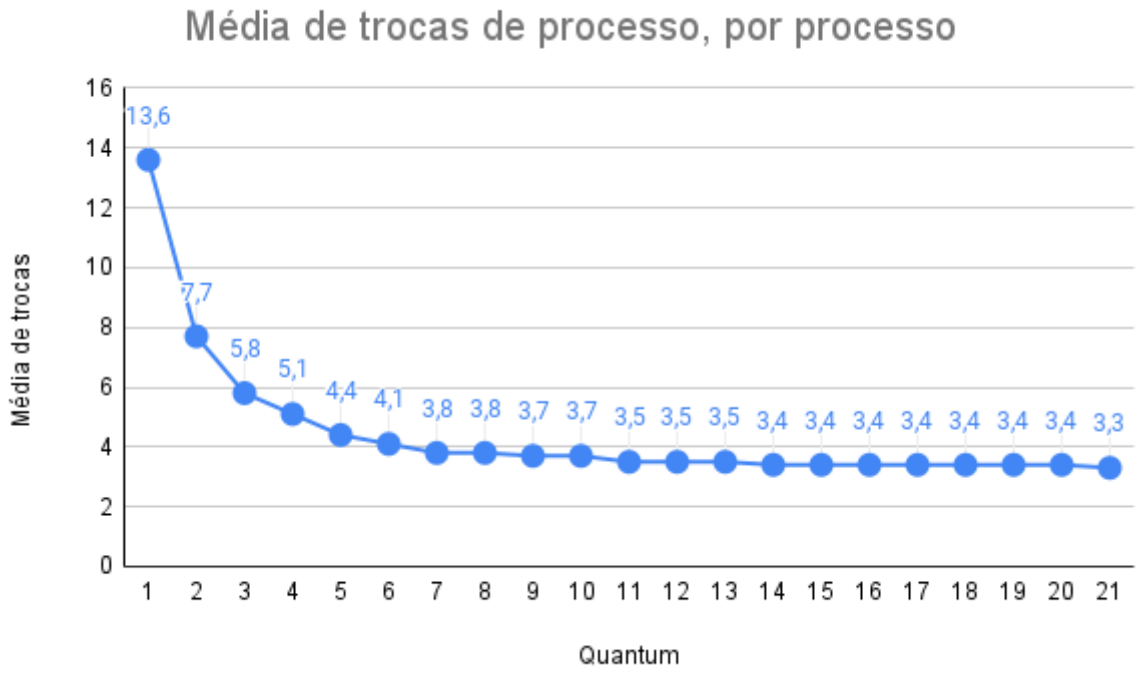


Gráfico 2: Média de instruções por quantum.

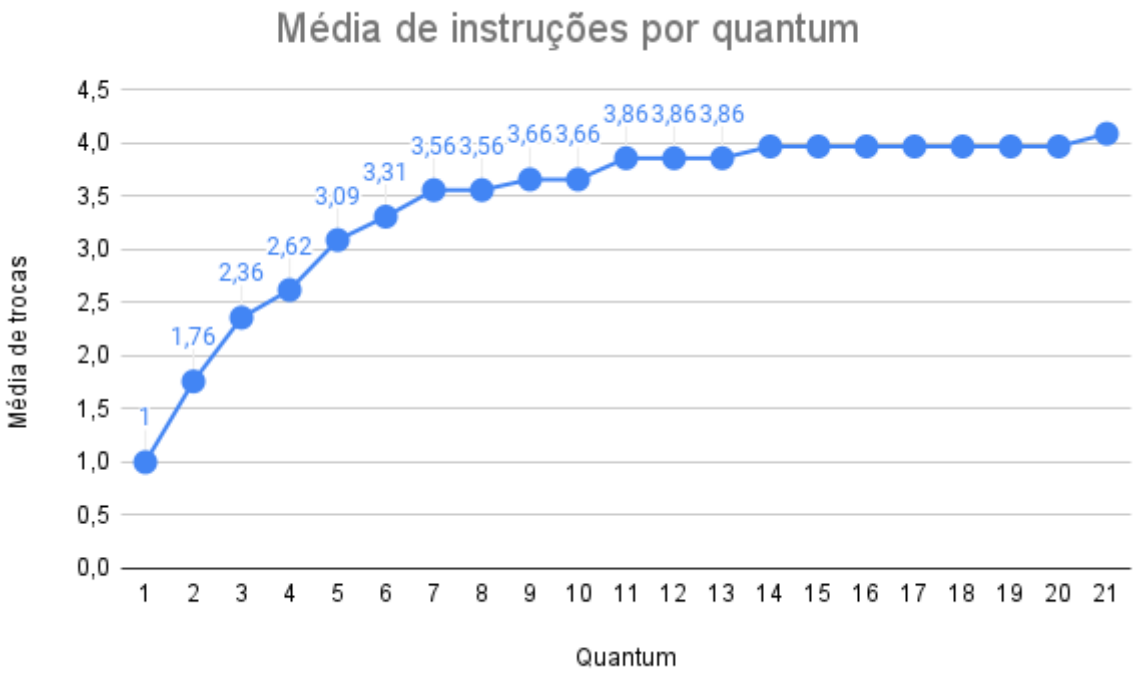
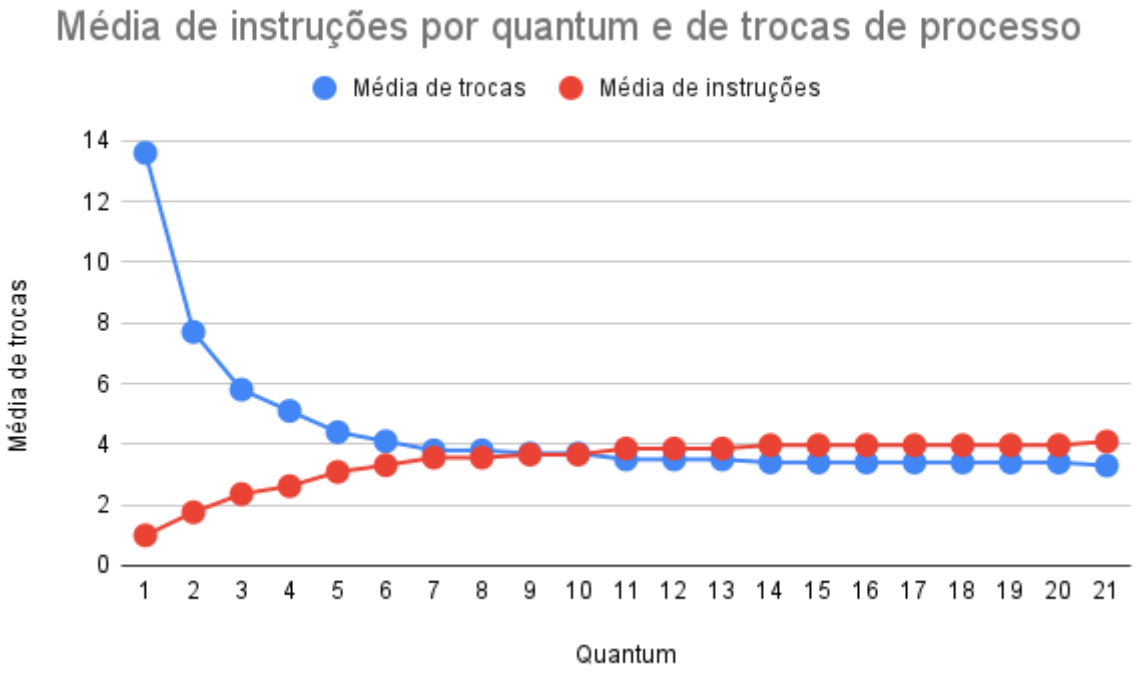


Gráfico 3: Médias de instruções e de trocas de processos por quantum.



ANÁLISE E CONCLUSÕES

No gráfico 1 é possível notar que a curva desenhada é muito semelhante à

uma curva logarítmica descendente, ou seja, conforme o número do quantum aumenta a média de trocas diminui, porém ao atingir um certo quantum, a curva se comporta de modo quase constante com o valor entre 6 e 7.

Já no gráfico 2, nota-se que o formato logarítmico se mantém porém de maneira ascendente. Fica claro também que ao se aumentar o quantum o número médio de instruções executadas também aumenta, tendendo a uma constância a partir de 11 quantum.

Unindo-se os 2 gráficos no gráfico 3 é evidente que ambos encontram estabilidade nas médias avaliadas entre 7 e 11 quanta, a partir desse valor o aumento de quantum deixa de gerar efeitos. Assim, para minimizar a quantidade de trocas de instrução evitando custo desnecessário ao processador em conjunto com uma não alta quantidade de quanta, acreditamos que o número ideal de quantum nestes experimentos seja entre 7 a 9 quanta.