Infraestrutura de Software
Implementação 2

Atenção:

- **Os arquivos de implementação, README e Makefile deverão ser salvos dentro de um diretório cujo nome será formado pelas iniciais do e-mail em minúsculas. Exemplo: considerando que o e-mail é est@cesar.school, o diretório será `est` Qualquer diretório enviado fora desse padrão será automaticamente descartado;**
- **O diretório será compactado em um arquivo `.tar`, e este deve ser disponibilizado para o professor via Form do Google Classroom da atividade. O nome do arquivo será formado pelas iniciais do e-mail em minúsculas. Exemplo: considerando que o e-mail é est@cesar.school, o arquivo a ser entregue será `est.tar` . Qualquer arquivo enviado fora desse padrão será automaticamente descartado;**
- **O nome do arquivo referente ao relatório deve ser formado pelas iniciais do e-mail em minúsculas do/a estudante e serem submetidas no Form da atividade com a extensão `.pdf`. Qualquer arquivo enviado fora desse padrão será automaticamente descartado;**
- **Identificada a cópia, seja com relação a outra/o estudante ou de alguma outra fonte, TODA a submissão será descartada.**

Projetando uma Memória Virtual

O exercício corresponde à implementação do *Designing a Virtual Memory Manager* apresentado na página P-51 do livro *Operating System Concepts, Silberschatz, A. et al*, 10a edição. Contudo, seguem algumas modificações:

- A implementação deverá ser aquela em que a memória física tem apenas 128 frames

- O programa deve ser implementado em C e ser executável em sistemas Linux, Unix ou macOS, com a compilação feita `Makefile`, através simplesmente do comando `make` via terminal, e retornar o arquivo com nome `vm` executável;

- Os frames na memória física devem ser preenchido do 0 ao 127, e quando a memória estiver cheia, aplicasse o algoritmo de substituição a fim de identificar qual frame será atualizado;

- Deve-se implementar dois algoritmos de substituição de página, a saber `fifo` e `lru`, e dois para substituição da TLB, também `fifo` e `lru`;

- O primeiro argumento por linha de comando será um arquivo de endereços lógicos (similar ao `address.txt` anexado ao Classroom), o segundo argumento será o tipo de algoritmo a ser utilizado para substituição da página (`fifo` ou `lru`), e o terceiro argumento o algoritmo a ser utilizado na substituição da TLB (`fifo` ou `lru`). Por exemplo, a chamada:

```
./vm address.txt lru fifo
```

  indica que o algoritmo de substituição da página será o `lru` e da TLB o `fifo`.

- A busca pela referência na TLB deverá ser realizada com uma `thread` para cada entrada da TLB a fim de simular uma otimização da busca;

- O arquivo de saída será denominado como `correct.txt`, seguindo a mesma formatação do que foi anexado na atividade do Classroom, que no caso utilizou o `fifo` tanto na substituição da página quanto na TLB;

- O programa não deve falhar se encontrar um erro, precisando checar todos os parâmetros antes de aceita-los. Assim, seu programa deve apresentar um mensagem de erro coerente e tratá-la, encerrando a execução quando: (1) o número e o tipo de argumentos durante a execução não estiverem de acordo com o descrito acima (2) se o arquivo de endereços não existir, não puder ser aberto ou encontrar-se numa formatação diferente daquela indicada no exemplo anexado.

O que deve ser submetido pelo Form do Classroom:

- relatório no formato `.pdf`, descrevendo o que foi feito, explicando, com detalhes suficientes para que o professor consiga entender, como as implementações foram feitas e o que foi testado. Apresente as dificuldades ao longo do processo e o que não conseguiu implementar, justificando se possível. O formato será livre, e inclua figuras, tabelas, gráficos, ou o que achar necessário para o entendimento. **Mesmo que não consiga implementar toda a atividade, indique o que foi realizado e como o professor posso testar.**
- os arquivos com o código, os testes usados, scripts, etc;
- um breve `README` descrevendo todos os arquivos `.c` utilizados, como executar, compilar e testar o programa, além de especificar em que sistema operacional foi implementado;
- incluir o `Makefile` para compilação, limpeza dos arquivos compilados e execução dos testes caso existam.

Eis uma sugestão para ordem da implementação com os respectivos pesos:

1. Tradução dos endereços (5%)
2. Fluxo sem TLB e tabela de páginas com 256 frames (10%)
3. Fluxo sem TLB e tabela de páginas com 128 frames substituindo com FIFO (10%)
4. Fluxo sem TLB e tabela de páginas com 128 frames substituindo com LRU (15%)
5. Fluxo com TLB sem thread substituindo com FIFO (5%)
6. Fluxo com TLB sem thread substituindo com LRU (5%)
7. Fluxo da TLB com thread (10%)

Pontuação

README + Makefile - 10% (proporcional ao que foi implementado)
Relatório - 20% (proporcional ao que foi implementado)
Tratamento de erro - 10%
Implementação - 60% (ver acima sugestão de ordem para implementação)