BCC266 – Organização de Computadores (2022-01)

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (www.decom.ufop.br/)

Trabalho Prático 2 (TP2) - 10 pontos, peso 3,5.

- Data de entrega: 14/02/2023 até 13:00. O que vale é o horário do *Moodle*, e não do *seu*, ou do *meu* relógio!!!

- Clareza, identação e comentários no código também vão valer pontos. Por isso, escolha cuidadosamente o nome das variáveis e torne o código o mais legível possível.

- O padrão de entrada e saída deve ser respeitado exatamente como determinado no enunciado. Parte da correção é automática, não respeitar as instruções enunciadas pode acarretar em perda de pontos.

- Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.

- A avaliação considerará o tempo de execução e o percentual de respostas corretas.

- Eventualmente serão realizadas entrevistas sobre os estudos dirigidos para complementar a avaliação;

- O trabalho é em grupo de até 3 (três) pessoas.

- Entregar um relatório.

- Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.

- Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.

- Códigos ou funções prontas específicas de algoritmos para solução dos problemas elencados não são aceitos

- Não serão considerados algoritmos parcialmente implementados.

- Procedimento para a entrega:.

1. Submissão: via ***Moodle***.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via ***Moodle***.
6. Você deve submeter os arquivos *.h*, *.c* e o *.pdf* (relatório) na raiz do arquivo *.zip*. Use os nomes dos arquivos *.h* e *.c* exatamente como pedido.
7. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.

- Bom trabalho!

Hierarquia de Memórias

Neste trabalho, o aluno entrará em contato com sistemas de memória, particularmente com o sistema cache.

O aluno irá adicionar um sistema de cache em três níveis ao TP1, assim como ilustra a Figura 1. A máquina implementada já possui dois níveis de cache.

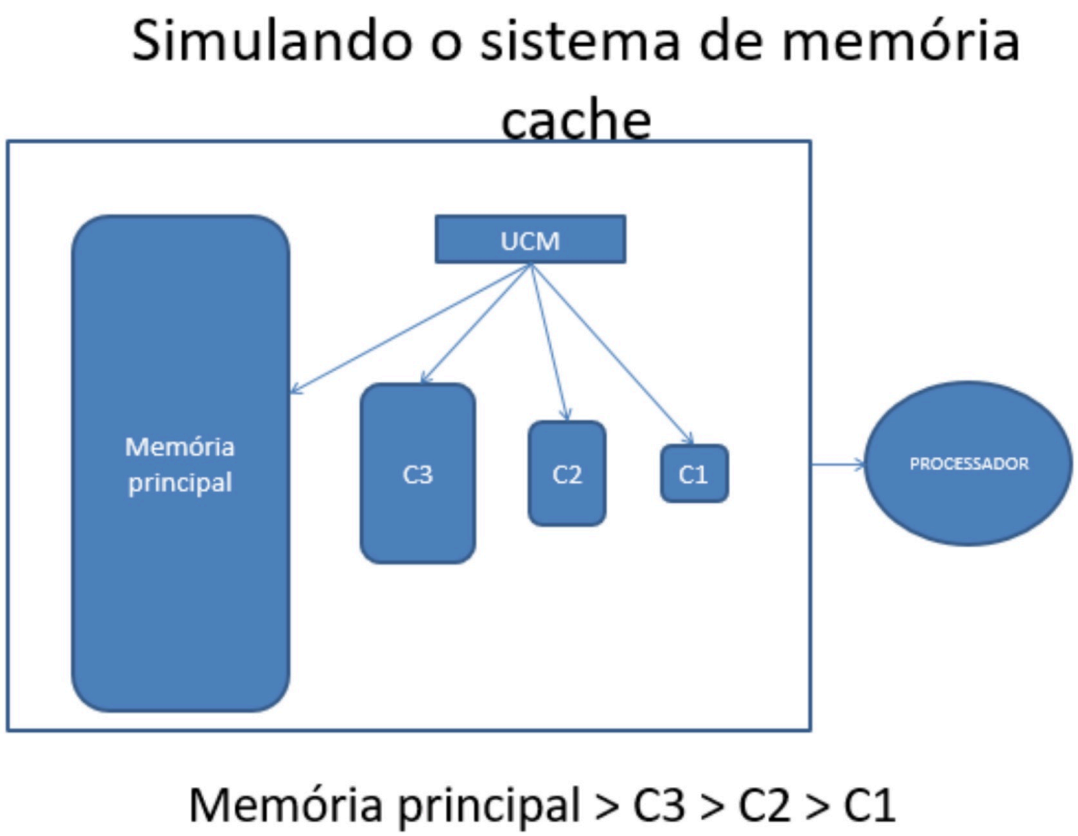

Figura 1: Sistema de memória cache

Assim como no TP1, as memórias possuem conteúdos denominados palavras e estas são do tipo inteiro (se quiser, pode usar outros tipos suportados pela linguagem escolhida para fazer o TP). A memória principal é particionada em blocos e a memória cache é particionada em linhas, conforme ilustra a Figura 2.

A sua tarefa é simular o mapeamento associativo ou o mapeamento associativo em conjunto para a troca de linhas (e consequentemente palavras) entre as caches e a memória principal (RAM). Não pode ser o mapeamento direto disponibilizado pelo professor. Este serve apenas como base para a construção dos demais.

Para substituição de linhas de cache, use pelo menos duas das políticas explicadas pelo professor, como por exemplo FIFO (*first-in first-out*), LFU (*last frequent used*) e LRU (*last recent used*).

Por fim, utilize o gerador de instruções, disponível no site da disciplina para as linguagens Java, C++ e Python. Tais geradores simulam as repetições de instruções necessárias para o efetivo resultado das memórias cache.

Além das tarefas de implementação, você deve realizar experimentos variando o tamanho das caches e avaliando a quantidade *cache hit* e *cache miss*. Na forma de tabelas (Tabela 1 por exemplo), ilustre *cache hit* e *cache miss*, assim como tempo hipotético de execução do programa, por exemplo. Altere os tamanhos de cache, o número de caches , o nível de repetição de instruções e as políticas de substituição.

Diferente do TP1, no TP2 as memórias (L1, L2, L3 e RAM) não serão mais vetores de inteiros simples, mas sim vetores de blocos no caso da RAM e vetores de linhas no caso das memórias cache. Cada bloco ou cada linha é um vetor de inteiros com um número definido de palavras.

Nessa nova máquina, o aluno terá que mudar o acesso à RAM, pois agora há também o acesso às memórias caches. As operações de manipulação de memória serão encapsuladas em uma função chamada MMU (*Memory Management Unit)* ou em português, Unidade de Controle de Memória). Não há mais um acesso direto à memória, mas sim pela MMU.

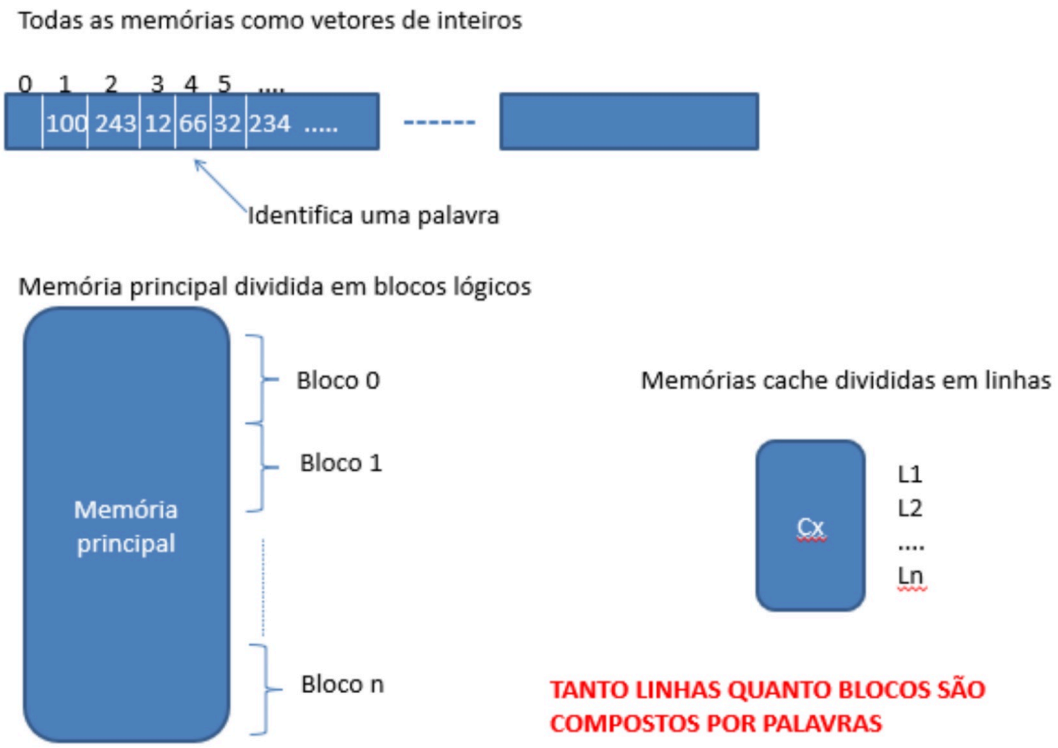

Figura 2: Sistema de memória em blocos

	Cache 1	Cache 2	Cache 3	Taxa C1 %	Taxa C2 %	Taxa C3 %	Taxa RAM %	Taxa de Disco %	Tempo de execução (unidade hipotética)
M1	8	16	32						
M2	32	64	128						
M3	16	64	256						
M4	8	32	128						
M5	16	32	64						

Tabela 1: Exemplo dos resultados

Imposições e comentários gerais

Neste trabalho, as seguintes regras devem ser seguidas:

- Seu programa não pode ter *memory leaks* (caso a sua linguagem tenha esse problema), ou seja, toda memória alocada pelo seu código deve ser corretamente liberada antes do final da execução. (Dica: utilize a ferramenta *valgrind* para se certificar de que seu código libera toda a memória alocada)
- Um grande número de *Warnings* ocasionará a redução na nota final.
- Implementações diferentes do que foi solicitado serão desconsideradas.

O que deve ser entregue

- Código fonte pode ser em C/C++, Java ou Python (bem identado e comentado).
- Documentação do trabalho (relatório). A documentação deve conter:

1. **Implementação:** descrição sobre a implementação do programa. Não faça "print screens" de telas. Ao contrário, procure resumir ao máximo a documentação, fazendo referência ao que julgar mais relevante. É importante, no entanto, que seja descrito o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.
2. **Impressões gerais**: descreva o seu processo de implementação deste trabalho. Aponte coisas que gostou bem como aquelas que o desagradou. Avalie o que o motivou, conhecimentos que adquiriu, entre outros.
3. **Análise**: deve ser feita uma análise dos resultados obtidos com este trabalho. **Você também deve analisar a ordem de complexidade do seu código.**
4. **Conclusão**: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
5. **Formato:** PDF ou HTML.

Como deve ser feita a entrega

Verifique se seu programa compila e executa na linha de comando antes de efetuar a entrega. Quando o resultado for correto, entregue via *Moodle* até a 14/02/2023 até 13:00 um arquivo **.ZIP**. Esse arquivo deve conter: (i) os arquivos *.c* e *.h* utilizados na implementação, (ii) instruções de como compilar e executar o programa no terminal, e (iii) o relatório em **PDF** (não esqueça de colocar seu nome e do grupo no relatório pois o *Moodle* renomeia os arquivos enviados).

Detalhes da implementação

O código-fonte deve ser modularizado corretamente. A separação das operações em funções e procedimentos está a cargo do aluno, porém, não deve haver acúmulo de operações dentro uma mesma função/procedimento.