BCC266 – Organização de Computadores (2022-02)

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

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

- Data de entrega: 21/03/2023 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!

Memória externa e interrupções

Neste trabalho, o aluno entrará em contato com sistemas de memória, adicionando a memória externa e tratamento de interrupções.

O TP2 continua válido, portanto o sistema de cache com três níveis ainda se mantém. O mapeamento associativo ou associativo em conjunto feito para o TP2 também se mantém. No TP3, haverá a chance da palavra de um bloco de memória não ser encontrada na memória principal (RAM), portanto a memória externa vai ser considerada como último nível, conforme ilustra a Figura 1. Com isso, o aluno irá adicionar a memória externa ao TP3.

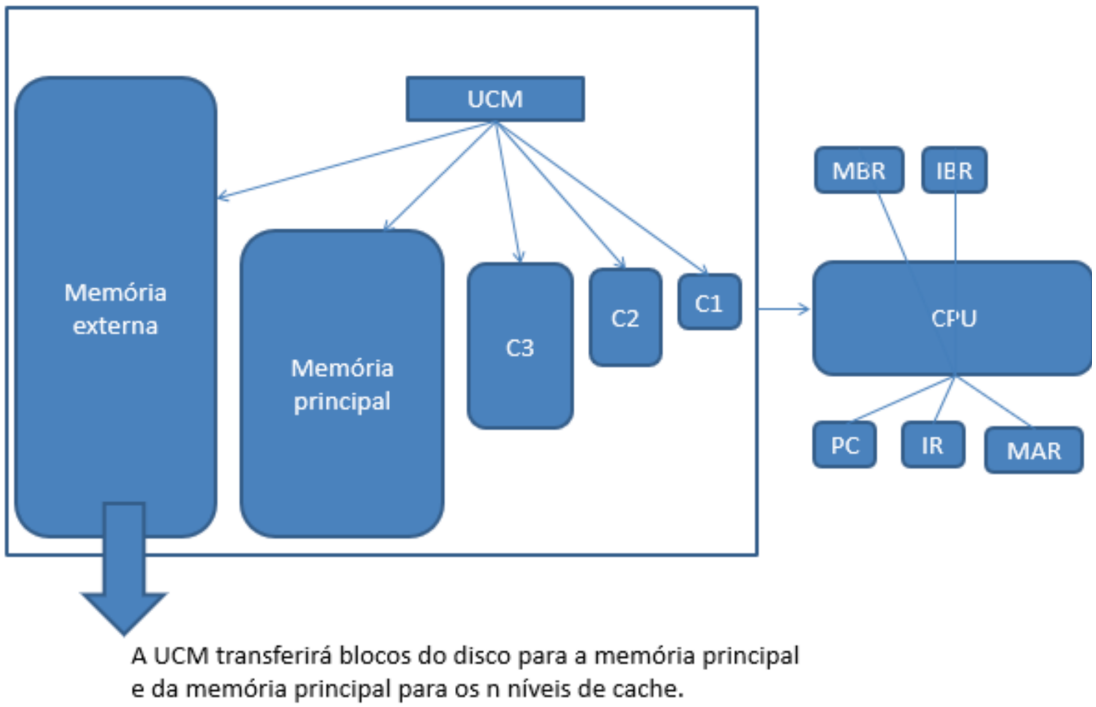

Figura 1: Sistema de memória (UCM = MMU).

Assim como no TP2, as memórias possuem conteúdos denominados palavras e estas podem ser do tipo inteiro, *bool*, byte ou qualquer outro suportado pela linguagem escolhida para fazer o TP. No TP2, a memória cache é particionada em linhas e a memória principal em blocos, para facilitar a construção do TP3, a memória externa também deverá ser particionada em blocos. No mundo real, o particionamento das caches em linhas, da memória principal em blocos e, por fim, da memória externa em trilhas e setores, se difere substancialmente um do outro.

A memória externa pode ser modelada como um único arquivo binário, permitindo facilmente achar um bloco com 4 palavras internas sem ter a necessidade de navegar por todo o arquivo toda vez que for necessário acessar o HD (acesso aleatório em arquivos - link). Outra possibilidade é modelar a memória externa como vários arquivos texto, onde cada arquivo representa um bloco com 4 palavras. Neste caso, o nome do arquivo é o endereço do bloco, portanto cada bloco é também facilmente encontrado.

Além de adicionar a memória externa, no TP3 o aluno deverá adicionar o conceito de interrupção na máquina sendo desenvolvida. Para isto, a cada execução de uma instrução a máquina verifica se há uma interrupção a ser tratada. O aluno deverá adicionar uma probabilidade de ocorrer uma interrupção após a interpretação de uma instrução do programa corrente, seja ele o aleatório, o gerado pelo GERADOR de instruções, o de multiplicação ou qualquer outro. Havendo uma interrupção, a máquina interrompe a interpretação do programa corrente, salva o contexto e inicializa a interpretação do programa denominado TRATADOR DE INTERRUPÇÃO. Após o término da interpretação de todo o TRATADOR DE INTERRUPÇÃO, a máquina volta a onde parou no programa corrente e prossegue normalmente até que uma nova interrupção ocorra. Vejamos em linhas gerais no Algoritmo 1 o funcionamento da máquina ao interpretar o programa corrente e o TRATADOR DE INTERRUPÇÃO (TI).

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*

Algorithm 1 Exemplo de funcionamento da máquina com tratamento de interrupção.

```
1: while há instruções no programa corrente do
2:     obtém uma instrução;
3:     switch (tipo de instrução) do
4:         soma { faz soma }
5:         sub { faz subtração }
6:         etc....
7:     end switch
8:     verifica se há interrupção;
9:     se há, salva o contexto;
10:    carrega as instruções do TI;
11:    continua interpretando normalmente as demais instruções....
12: end while
```

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.

	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 21/03/2023 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.

Você deve implementar:

- Adição da memória externa.
- Adição de um tratador de interrupção.