DCC006: Organização de computadores I **(Entrega: 04/11/2020)**

Trabalho Prático #3

Professor: Omar Paranaíba Vilela Neto *Monitor:* Pedro Arthur R. L. Silva

Antes de começar seu trabalho, leia todas as instruções abaixo.

- O trabalho pode ser feito em grupos compostos por até 3 alunos.
- Cópias de trabalho acarretarão em devida penalização às partes envolvidas.
- Entregas após o prazo serão aceitas, porém haverá uma penalização. Quanto maior o atraso maior a penalização.
- O objetivo desse trabalho é praticar alguns dos conceitos aprendidos sobre memória em arquitetura de computadores. Vocês devem fornecer um programa que simule o comportamento dos componentes que serão descritos logo mais. Você pode utilizar as seguintes linguagens de programação para a conclusão desse trabalho: *C*, *C++*, *Python*, *Java* e *Verilog*, caso queira utilizar outra linguagem, entre em contato com o monitor da disciplina antes.
- Você deve entregar um único arquivo zip, contendo a sua implementação da tarefa pedida, um *script bash* (extensão *.sh*) para execução do seu programa e um arquivo de texto (extensão .txt) como o nome e matrícula de todos os integrantes do grupo.
- O seu programa deve ser capaz de ser executado em um computador com as seguintes especificações:
  - Memória RAM: 8GB
  - Processador: Intel Core i5-7200U CPU @ 2.50GHz
  - Sistema Operacional: Ubuntu 20.04
- **Cada grupo deve fazer somente uma submissão**. Ou seja, cada grupo terá um aluno responsável por fazer a submissão do trabalho no *Moodle*.

Esse trabalho consiste na implementação de uma pequena hierarquia de memória. Nessa hierarquia, existe uma CPU que irá realizar operações de leitura e escrita em um subsistema de memória. Esse subsistema consiste em uma pequena memória cache e uma memória de dados.

A CPU pode disparar operações de escrita e leitura na memória. Operações de leitura enviam um endereço que desejam acessar, esse endereço é passado para a cache que retorna o dado caso um *hit* ocorra, ou busca um bloco na memória de dados caso um *miss* ocorra.

Nas operações de escrita, recebe-se um endereço e um dado, primeiro atualiza-se o bloco correspondente àquele endereço na cache e marca-se tal bloco como "sujo". Quando surge a necessidade de substituir um bloco "sujo" por um novo bloco na cache, então, primeiro atualizamos a memória de dados com os dados do bloco "sujo", e depois atualizamos a cache com os dados do novo bloco.

Para implementar tal sistema, faça as seguintes considerações:

- Sua memória de dados consegue armazenar 1024 palavras de 32 bits.
- Sua cache tem a capacidade de armazenar 64 blocos. Cada bloco da cache contém 16 *bytes*, que correspondem a 4 palavras de 32 bits, que resultam em 128 bits no total.
- Sua cache utiliza Mapeamento Direto para alocar os blocos.
- Para operações de escrita na memória você deve utilizar a técnica de *Write Back*.
- Os endereços que a CPU fornece contém 32 bits.

Para que seja possível testar o seu sistema, o seu programa deve ser capaz de ler um arquivo em que cada linha representa uma requisição da CPU, contendo as seguintes informações:

- Endereço N para acesso à memória. (0 <= N <= 2^{12})
- Tipo da operação: 0 para leitura, 1 para escrita
- Dado para operação de escrita. Um dado contendo 32 bits a ser escrito na memória.

Abaixo, segue um exemplo de tal arquivo:

```
5 1 00000000000000000000000000000101
5 0
12 1 00000000000000000000000000010010
25 0
```

A saída de seu programa deve ser outro arquivo, nomeado *result.txt*, com o seguinte formato:

```
READS: 2
WRITES: 2
HITS: 1
MISSES: 1
HIT RATE: 0.5
MISS RATE: 0.5

5 1 00000000000000000000000000000101 W
5 0 H
12 1 00000000000000000000000000010010 W
25 0 M
```

No resultado, note que utilizamos a sigla W para indicar que a operação é uma escrita, H para quando um *hit* ocorreu para aquela operação e M se foi um *miss*.