# Fundamentos de Sistemas de Operação

## MIEI 2019/2020

## Homework Assignment 2

### Deadline and Delivery

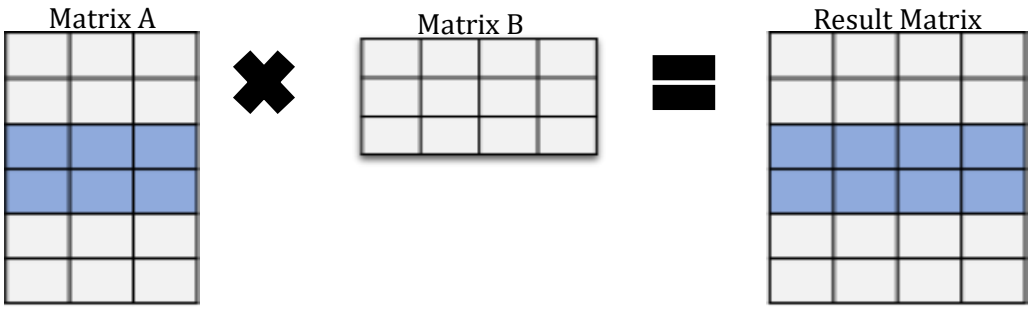
This assignment is to be performed ***individually*** by each student – any detected frauds will cause failing the discipline. The code has to be submitted for evaluation via the Mooshak system (`http://mooshak.di.fct.unl.pt/~mooshak/`) using each student's individual account -- the deadline is ***23h59, November 6th, 2019***.

### Description

The goal of this assignment is twofold. First, your program implements the multiplication of two matrices like in TPC1 but this time using the Pthreads library. Second, your program has to write, on the standard output, the average value of the result matrix.

As you recall, matrix multiplication may be a computationally heavy process and so you have to use multiple threads to carry out the operation. Accordingly, the initial thread (let's call it *main thread*) creates a given number of threads (let's call them *worker threads*) and assigns a task to each one, namely to comput part of the matrix multiplication.

The algorithm for this parallel matrix multiplication is quite simple. When computing A x B, each worker accesses a subset of the A's rows and the entire matrix B. This data is used to compute the part of the result matrix assigned to the worker thread, as illustrated in the next figure.




Given that this is an assignment of an Operating Systems' course, you will have to take advantage of your knowledge on thread creation and control. Data has to be shared between the master and its workers so that each worker calculates only its part of the matrix. Together they cooperate to produce the result matrix and to calculate the total average value in it. In this case, one of the workers is responsible to calculate de average value meaning that it is necessary to wait for all threads to finish their calculations, and only after that the average value may be calculated. Finally, the master outputs the result matrix and its total average value.

The behaviour of the master may be given by the following pseudo-code:

```
Read or generate the input matrices
Allocate space for the final result matrix
Define the rows associated with each worker
Launch all workers

Wait that all workers
       fill the result matrix, and
       calculate the total average value in it
Write the result matrix and the average value
```

In turn, the behaviour of each worker is:

```
Identify the rows in matrix A it has to access
Multiply rows in A to the full matrix B and fill the result rows in the
 final matrix B.
Cooperate with the other threads to calculate the total sum in
 the result matrix; a particular thread calculates the average value.
```

## Work to do

To allow for you to focus your attention on the abovementioned Operating Systems concepts, we already supply you with a codebase (downloadable from CLIP) with part of the work done. This codebase features 5 files, some of which you already know from TPC1:

- `main.c` is the file that starts the application and features the functions to implement.
- `matrix.h` and `matrix.c` provide struct `matrix` and a set of functions that you apply over matrices, such as obtaining a submatrix or performing a matrix multiplication.
- `log.h` and `log.c` provide a function to log messages emitted by the workers. The implementation writes these messages to the standard output.

To complete the given implementation, you will only have to work on file `main.c` and **this is the only file that you must submit to Mooshak**. To complete the assignment, you must:

- implement function `par_matmul_average` that launches the threads to perform the desired computation.
- Implement function `worker_function` that is the one that defines the behavious of each thread,

## Compile and Run the Application

To compile simply type

- `make`

To run you must supply the number of rows and columns of the first matrix (matrix A) and the number of columns of the second matrix (matrix B), and the number of workers. The matrices are generated by the application use pseudo-random numbers. You may supply the seed to the pseudo-random number. The command line is thus:

- `parmatmul nrows1 ncols1 ncols2 nworkers [seed]`

To use 5 workers to multiply a matrix of 10x20 with a matrix of 20x4, you need to type:

- `parmatmul 10 20 4 5`