SE assignment **Univ. Rennes 1**

Cache Access Optimization

We will explore the use of cache memory. In the following exercises, we will suppose that the cache memory is unique (and not separated in L1, L2 and L3 cache levels).

Warm-up exercise

The following algorithm computes the sum of the n elements of an array `v` of integers:

```
int sum = 0.0;
int n = v.length;
for (int i = 0; i < n; i++)  sum = sum + v[i];
```

Supposing that we have only one cache and that the cache chunk (or block) corresponds to `n/10`, how many *hits* and *misses* can you identify during the execution of the algorithm?

Main exercise

During the lecture, we have analyzed the memory access for a matrix-by-vector algorithm, with two possible representations for the matrix. We will consider now the more complex situation where we need to multiply two $n \times n$ matrices A and B. The standard algorithm for matrix-by-matrix multiplication is as follows:

```
for (int i = 0; i < n; i++)
   for (int j = 0; j < n; j++)
      C[i][j] = 0.0;
      for (int k = 0; k < n; k++)
         C[i][j] = C[i][j] + A[i][k]*B[k][j];
```

We are going to see whether we can optimize the access to the memory as long as the matrices A and B are concerned; we won't attempt any kind of access optimization for the resulting matrix C.

Question 1

Suppose that we can choose two distinct representations for the two matrices A and B. For the standard matrix-by-matrix algorithm, what are the two distinct representations resulting in the best performances in terms of cache hits? And what if we subsequently swap the two representations chosen for A and B?

Question 2

Programmers generally prefer to have a common data representation for all the "objects" of the same class. In order to have a clean and proper code, it is necessary therefore to fix one unique data representation for the two matrices A and B. Please select one of the two representations considered above and analyze the cache access for the standard matrix-by-matrix algorithm in terms of cache misses and hits in the given situation. You can present your analysis in a graphical way with an explanatory drawing.

Question 3

You should therefore remember how the data are able to flow from the main memory to the cache memory. When a specific element of one of two matrices is requested, this is not the only elemeent that is loaded in the cache. But can we exploit this mechanism to get a better (faster) access to the main memory? Please discuss the matter with your neighbor, and when necessary, with your teacher.

Question 4

Suppose that m is the cache block size (smaller than n). We are going to try out a very particular approach. Instead of working on the rows or on the columns of the two matrices, we will consider now *squared* sub-matrices of A and B. In order to simplify our calculations, we suppose that these squared sub-matrices have all size $m \times m$, and that m is a multiple of n. Can you rewrite the standard matrix-by-matrix algorithm (see above) in a higher-level language so that it can act on such sub-matrices, instead of working directly on the matrix elements?

Question 5

In your previous high-level code, you should have instructions for performing operations on matrices. The idea now is to replece such high-level instructions with their C/Java low-level equivalent (i.e. we want to directly act on the matrix elements!). *Hint*: you can make reference (again!) to the standard matrix-by-matrix algorithm!

Question 6

Perform an analysis on the number of cache hits and cache misses for the algorithm you have just written. Recall we supposed that m is the cache chunk size.