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).

Exercise 1

The following algorithm computes the sum of the elements of an n-vector `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?

Exercise 2

Your employer is asking you to revise the software you have developed in order to improve its performance in terms of time. Your code contains the following lines:

```
for (int j = 0; j < 10; j++)
   for (int i = j; i < n; i = i + 10)
      r[i] = alpha[j]*r[i];
```

Can you visualize with a drawing how this piece of code makes access to the cache memory? Is there any way to optimize these instructions?

Exercise 3

During the lecture, we have studied the memory access for a matrix-by-vector algorithm, with two possible representations for the matrix. We will study 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];
```

Please follow the points below.

1. Suppose that we can choose two distinct representations for the two matrices, one representation for A, and another representation for B. For the standard matrix-by-matrix algorithm, what are the two distinct representations that could give us the best performance in terms of cache hits? And what if we subsequently swap these two representations?

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.

3. You know how the data are able to flow from the main memory to the cache memory. You know that, when a specific element of one of two matrices is requested, the elements stored in consecutive memory addresses are also loaded in the cache. With this idea in mind, please try to devise a new algorithm for the matrix-by-matrix multiplication where the cache use is optimized by reducing the number of cache misses. Please answer to the questions below to be guided towards the solution:

   - It is inevitable to load data in the cache memory that cannot be used immediately. However, is there any way to keep these unused data in the cache until they will be necessary for the calculations?
   - Suppose that m is the cache block size (smaller than n), and suppose for simplicity that it is a multiple of n. What sub-matrix "shape" could satisfy the following two properties? *Property* 1: when the sub-matrix is extracted from A, and another with the same shape is extracted from B, the multiplication of the two sub-matrices is a sub-matrix of C that preserves the shape; *Property* 2: when loading the two sub-matrices in the cache, the number of cache misses and hits is the same for A and B.
   - Before writing the algorithm in a low-level programming language, suppose that our language allows us to work directly with the sub-matrices defined above. Write the pseudo-code by following the main steps of the standard matrix-by-matrix algorithm.
   - Finally, identify the operations in your previous high-level code that are performed on sub-matrices, and replace them with a more explicit low-level code where the required basic arithmetic operations on the matrix elements are given. *Hint*: you can make reference (again!) to the standard matrix-by-matrix algorithm reported in the previous page.