HOMEWORK 1: MATRIX TRAVERSAL (10 POINTS)

Create a large matrix of *contiguous* memory elements which does not fit in the L2 cache of a machine.

1. Traverse the matrix first raw-wise, then column-wise, and measure the execution times in both cases; **(3 points)**
2. Use the *valgrind* tool to measure the number of cache misses in each case and explain the difference in execution time; **(4 points)**
3. Solve the problem for both C and FORTRAN programs and explain the difference that you observe. **(3 points)**

Important: Do not use any compiler optimisations.

Matrix traversal in FORTRAN:

```
program matrix
      implicit none
      integer, parameter :: dim = 2048
      integer i, j
      real*8 a(dim, dim)
      do 10 i = 1, dim
            do 10 j = 1, dim
                  a(i,j) = 0.01 * i + j
      10  continue
end
```

HOMEWORK 2: ALIGNED ARRAYS (10 POINTS)

Consider the following program:

```
#define max 1024 * 1024

float a[max], b[max], c[max], d[max];

for(i = 0; i < max; i++)

   a[i] = b[i] + c[i] * d[i];
```

1. Explain the performance of this program by measuring the number of *L1* and *L2* cache misses on a machine with 8 MB of two-way set associative L2 cache size and 128 cache line size using *valgrind*; **(3 points)**
2. Modify the program to make it faster. Analyse the execution times and *L1* and *L2* cache misses in modified program compared to the original one using *valgrind*; **(4 points)**
3. Extend the original program and observe the difference in execution time on a real machine (without using *valgrind*). **(3 points)**

Important: Do not use any compiler optimisation flags (e.g. –O).