# Tema 2 ASC – Inmultirea Eficienta a Matricelor in C/C++

**2017**

**Responsabili:**
Cosmin-Gabriel Samoila
Emil Slusanschi
Silvia Stegaru
Mihaela Gaman

## Cerinta:

Se da o implementare in Python ce implementeaza urmatoarea operatie cu matrici:

**C = alpha * A * B + beta * C**

- A, B, C – matrici bidimensionale de double
- alpha, beta – constante double

Codul in Python(solver.py) este implementat utiliziand Matrix Matrix Multiply din BLAS: http://www.netlib.org/lapack/explore-html/d7/d2b/dgemm_8f_source.html

## Observatii:

- Structura detaliata a inputului, outputului si a testelor o gasiti in README.md si in solver.py / utils.py
- Puteti rezolva atat in C cat si in C++. Trebuie sa aveti un makefile ce genereaza un binar la comanda **make** folosind **gcc** sau **g++** (v 5.4).
- **Atentie:** nu realizati implementarea finala bazandu-va pe feature-uri din alte compilatoare (ex: **icc**) pentru ca nu o sa fie disponibil niciun alt compilator in afara de gcc si g++ in momentul testarii.
- Va este oferit un schelet de cod in C/C++ ce contine parsarea fisierelor de configurare. Voi trebuie sa:
  1. Cititi matricile A, B, C (stiti dimensiunile M,N,K si numele fisierelor din struct test)
  2. Implementati dgemm astfel incat sa obtineti rezultatele corecte pentru orice combinatie de configuratii pentru A si B (transpuse / netranspuse).
  3. Scrieti rezultatele obtinute in fisierul de output corespunzator testului.
- Va sunt oferite doar cele 4 teste de functionalitate NT / TN / TT / NN (semnificand AB – transpusa/netranspusa) cu dimensiuni M=N=K=1000.
- Implementarea nu trebuie sa dureze mai mult de 3 minute pentru a primi punctajul temei si trebuie sa dureze mai putin de 45 de secunde pentru a primi bonus.
- Aveti puse la dispozitie input_generator.py si tester.py prin care puteti sa va generati teste de diverse dimensiuni si sa testati implementarea. Este util sa va generati fisiere de input de aceleasi dimensiuni cu testele 1-4 si sa verificati timpul total de rulare pentru a fi siguri ca nu depasiti cele 3 minute.
- Daca rularea dureaza mai mult de 3 minute insumat pe cele 4 teste de functionalitate + testele 1-4, pierdeti **TOT** punctajul.

## Punctaj:

1. 70p testarea automata (functionalitate / 30p; testele 1-4 / cate 10p pentru fiecare test)
2. 30p explicati rezultatele obtinute (10p) in readme (10p) si construiti grafice relevante (10p) pentru implementarea voastra (e.g. pornind de la scripturile gnuplot din laboratorul 4).
3. Bonus 20p – conditionat de partea 1 si 2 (nu veti primi punctajul daca implementarea voastra este incompleta/incorecta dar termina in mai putin de 45 de secunde) – implementarea trebuie sa rezolve toate testele in mai putin de 45 de secunde.