

# Multiprocesorski sistemi

Domaći zadatak 4  
Školska godina 2024/2025  
CUDA – osnove  
(10 poena)

## Uvod

Cilj zadatka je da studente obuči da samostalno razvijaju osnovne CUDA programe za izvršavanje na grafičkom procesoru.

## Podešavanje okruženja

Detaljna uputstva za instaliranje, podešavanje i prvo izvršavanje CUDA programa se mogu naći na adresi <http://developer.nvidia.com/nvidia-gpu-computing-documentation>. Po tom uputstvu podesiti okruženje za razvoj i kontrolisano izvršavanje (engl. debugging) CUDA programa na lokalnom računaru. Alternativno, koristiti CUDA (**nvcc**) na računaru **rtidev5.etf.rs**. Prevodilac se nalazi u direktorijumu: **/usr/local/cuda/bin/**.

## Izveštaj

Uz predati domaći zadatak (izvorne kodove) treba napisati i priložiti kratak izveštaj o izvršenoj paralelizaciji i dobijenim ubrzanjima u odnosu na sekvencijalnu verziju koda. Za svaki rešeni zadatak treba kratko opisati uočena mesta koja je moguće paralelizovati i način paralelizacije. Takođe, potrebno je dati logove izvršenog koda za sve test primere koji se izvršavaju i nalaze se u **run** datoteci i nacrtati grafike ubrzanja u odnosu na sekvencijalnu verziju. Na graficima je potrebno dati i rezultate poređenja različitih načina paralelizacije za isti broj niti, ukoliko postoje takvi zahtevi u okviru teksta zadatka. Šablon za pisanje izveštaja se nalazi u okviru sekcije za domaće zadatke predmetnog sajta.

## Zadaci

Svi programi treba da koriste GPU za bilo koju obradu. Smatrati da je broj GPU niti na nivou jednog bloka niti određen konstantom **NUM\_OF\_GPU\_THREADS**, čija je vrednost za sve zadatke 1024. Obezbediti da niti koje u nekom koraku nemaju posla na korektan način stignu do kraja tela CUDA jezgra.

Kod zadatka gde je to zahtevano, korisnik zadaje samo dimenzije nizova/matrica, a sve potrebne ulazne podatke generisati u operativnoj memoriji uz pomoć generatora slučajnih brojeva iz biblioteke jezika C, a zatim prebaciti u GPU memoriju. Generisani brojevi treba da budu odgovarajućeg tipa u opsegu od **-MAX** do **+MAX**, gde **MAX** ima vrednost 1024. Za sve zadatke je potrebno napisati ili iskoristiti zadataku sekvencijalnu (CPU) implementaciju odgovarajućeg problema koja će biti korišćena kao referentna (*gold*) implementacija prilikom testiranja programa.

Svaki program treba da:

- Generiše ili koristi već obezbeđene ulazne test primere.
- Kopira test primere u GPU memoriju i rezultat iz GPU memorije.
- Izvrši CUDA jezgro nad zadatim test primerom.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Ispiše vreme izvršavanja CUDA i sekvencijalne implementacije problema.
- Uporedi rezultat CUDA i sekvencijalne implementacije problema.
- Ispiše "**Test PASSED**" ili "**Test FAILED**" u zavisnosti da li se rezultat izvršavanja MPI implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Kod zadataka koji koriste realne tipove (**float**, **double**) tolerisati maksimalno odsupanje od  $\pm$ **ACCURACY** prilikom poređenja rezultata CPU i GPU implementacije. Smatrati da konstanta **ACCURACY** ima vrednost 0.01. **Prilikom rešavanja zadataka voditi računa da se postigne maksimalni mogući paralelizam.** Dozvoljeno je ograničeno preuređivanje dostupnih sekvencijalnih implementacija prilikom paralelizacije. **Ukoliko u nekom zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku i da nastavi da izgrađuje svoje rešenje na temeljima uvedene pretpostavke.**

Dostupne sekvencijalne implementacije se nalaze u arhivi **MPS\_DZ.tar.gz** koje se mogu preuzeti na adresi <http://mups.etf.rs/dz/2024-2025/>. Na **rtidev5.etf.rs** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: `wget http://mups.etf.rs/dz/2024-2025/MPS_DZ.tar.gz`

Raspakivanje: `tar xjvf MPS_DZ.tar.gz`

1. Paralelizovati program koji vrši jednostavno generalizovano množenje matrica u jednostrukoj preciznosti *Single precision floating General Matrix Multiply* (SGEMM). SGEMM operacija je definisana sledećom formom:

$$C \leftarrow \alpha \cdot A \cdot B + \beta \cdot C$$

Program se nalazi u datoteci **sgemm.cc** u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u **run** skripti.

Prilikom zadavanja izvršne konfiguracije jezgra, koristiti 1D rešetku (grid).

2. Paralelizovati program koji izračunava aproksimaciju integrala koristeći [Gauss-Laguerre kvadraturno pravilo](#). Po ovom pravilu, integral oblika:

$$\int_a^{\infty} e^{-b(x-a)} f(x) dx$$

se aproksimira sumom:

$$\sum_{i=1}^{order} \omega(i) \cdot f(x(i))$$

gde su  $\omega(i)$  težinski koeficijenti, a  $x(i)$  odgovarajuće čvorne tačke.

Program se nalazi u direktorijumu **gauss\_laguerre** u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u **run** skripti.

3. Paralelizovati program koji vrši statističku analizu prostorne distribucije posmatranih astronomskih tela korišćenjem tzv. *two point angular correlation function* (**TPACF**). Algoritam proračunava uglovne distance između svih parova tačaka (tela) sa ulaza i generiše histogram posmatranih distanci.

Program se nalazi u direktorijumu **tpacf** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **main.c** i **model\_compute\_cpu.c**. Analizirati dati kod i obratiti pažnju na algoritam za izračunavanje distanci unutar datoteke **model\_compute\_cpu.c**. Program testirati sa parametrima koji su dati u **run** skripti.