Multiprocesorski sistemi

Domaći zadatak 4
CUDA – osnove
(10 poena)

Uvod

Cilj zadatka je da studente obuči da samostalno razvijaju osnovne CUDA programe.

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 ili na sajtu predmeta pod nazivom CUDA Getting Started Guide (Windows) ili CUDA Getting Started Guide (Linux) u zavisnosti koji operativni sistem se koristi. Po tom uputstvu podesiti okruženje za razvoj i kontrolisano izvršavanje (engl. debugging) CUDA programa na lokalnom računaru. Alternativno, koristiti CUDA (**`nvcc`**) na računaru **`rtidev4.etf.rs`**.

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 256. Obezbediti da niti koje u nekom koraku nemaju posla na korektan način stignu do kraja tela CUDA jezgra.

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. Specifično, u 4. i 5. zadatku generisani podaci treba da budu u opsegu od 0 do 255. Za sve zadatke je potrebno napisati i sekvencijalnu (CPU) implementaciju odgovarajućeg problema koja će biti korišćena kao referentna (*gold*) implementacija prilikom testiranja programa. Svaki program treba da:

- Generiše 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.
- Uporedi rezultat CUDA i sekvencijalne implementacije problema.
- Ispiše vreme izvršavanja CUDA i sekvencijalne implementacije problema.
- Ispiše "**`Test PASSED`**" ili "**`Test FAILED`**" u zavisnosti da li se rezultat izvršavanja CUDA implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Kod zadataka koji koriste realne tipove (**`float`**, **`double`**) tolerisati maksimalno odsupanje od ±**`ACCURACY`** prilikom poređenja rezultata CPU i GPU implementacije. Smatrati da konstanta **`ACCURACY`** ima vrednost 0.001.

1. Sastaviti program koji kvadrira elemente dvodimenzionalne matrice celih brojeva.
2. Sastaviti program koji računa proizvod dve realne matrice proizvoljnih dimenzija. Izvršiti proveru da li se zadate dve matrice mogu pomnožiti.
3. Sastaviti program koji pronalazi najmanji i najveći element dvodimenzionalne matrice celih brojeva.
4. Sastaviti program koji računa 8-bitnu LRC kontrolnu sumu za zadati niz znakova (string). Kontrolna suma se najpre računa na nivou bloka od 1024 elementa koji se po potrebi dopunjuje nulama, a zatim se vrši XOR operacija nad svim dobijenim sumama da bi se dobila konačna suma na nivou celog stringa. Računanje kontrolne sume na nivou bloka se vrši sledećim sekvencijalnim kodom:

```
unsigned char calculateLRC(const unsigned char *data, unsigned int n) {
  unsigned char checksum = 0;
  for (int i = 0; i < n; i++)
    checksum = (unsigned char) ((checksum + data[i]) & 0xFF);
  checksum = (unsigned char) (((checksum ^ 0xFF) + 1) & 0xFF);
  return checksum;
}
```

5. Prethodni problem rešiti korišćenjem deljene memorije. Obezbediti što je moguće bolje performanse prilikom čitanja bloka podataka iz globalne memorije.

Važno: 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.