Arhitektura računara 2021/22 – Teme za drugi projektni rad

Napomena za izradu projektnih zadataka

Projektni rad se mora moći kompajlirati, izvršiti i testirati. Uz priloženi rad treba da se obezbijede i primjeri koji se mogu iskoristiti za testiranje programa.

Zadatak 2.1 – Tema po prijedlogu studenta

Tema po prijedlogu studenta, vezana za arhitekturu računara. Prema dogovoru sa predavačem.

Zadatak 2.2 – Simulator keš memorije

(10) U proizvoljnom programskom jeziku napisati pojednostavljen simulator keš memorije. Programu je potrebno proslijediti veličinu radne memorije, veličinu keš memorije i veličinu keš linije. Keš treba podrazumijevano da radi kao direktno-mapirani keš koji vrši zamjenu linija primjenom *least recently used* (LRU) algoritma. Keš memorija treba da imitira rad po *write-back* mehanizmu. Omogućiti da se interaguje sa programom slanjem naredbi za simulirani pristup memoriji ili funkcija koje će da vrše pristupe memoriji. Pri svakom pristupu, potrebno je evidentirati da li je došlo do keš pogotka (*cache hit*) ili promašaja (*cache miss*). Potrebno je voditi evidenciju o ukupnom broja pogodaka i promašaja.

(5) Omogućiti da keš memorija bude N-*way associative*, tako da korisnik može konfigurisati vrijednost za $N (N = 2^k)$.

(5) Implementirati i optimalni (*Bélády*) algoritam za zamjenu linija, uporediti ponašanje u odnosu na LRU pristup i dokumentovati rezultate.

(5) Modifikovati način pristupa memoriji tako da se upis i čitanje posmatraju kao različite vrste pristupa. Simulirati pristup memoriji od strane dva procesorska jezgra korištenjem niti. Svaka nit treba da pristupa sopstvenoj strukturi koja predstavlja lokalnu keš memoriju, pri čemu je radna memorija dijeljena. Pri tome, na proizvoljan način riješiti problem koherencije keš memorije i demonstrirati rješenje na primjerima u kojima navedene niti rade konkurentno i imitiraju pristupe memoriji.

Obezbijediti nekoliko primjera ili jediničnih testova kojima se demonstriraju funkcionalnosti iz stavki u specifikaciji projektnog zadatka.

Sve detalje koji nisu eksplicitno navedeni implementirati na proizvoljan način.

Pridržavati se:

- principa pisanja čistog koda i pravilnog imenovanja varijabli, funkcija, klasa i metoda
- konvencija za korišteni programski jezik

Zadatak 2.3 – Optimizacija algoritma

(15) U proizvoljnom programskom jeziku realizovati proizvoljan paralelizibilan algoritam koji vrši netrivijalnu obradu podataka. Algoritam odabrati na kursu predmeta. Analizirati, uporediti, dokumentovati i grafički predstaviti mogućnosti ubrzavanja datog algoritma korištenjem kompajlerskih optimizacija i bar 2 navedena pristupa:

1. (a) SIMD programiranje ili (b) optimizacije za keš memoriju
    - Za (b) je potrebno izvršiti mjerenja (npr. upotrebom *cachegrind* alata) i pokazati da keš optimizacija stvarno doprinosi keš performansama (procentu keš pogodaka)
    - Za (b) je dozvoljeno da se kao inicijalni algoritam iskoristi varijanta algoritma sa lošijim keš performansama (bez dodavanja nepotrebnog koda)
2. Paralelizacija na višejezgarnom procesoru (npr. OpenMP ili sopstveno rješenje)
3. Distribuirano procesiranje (npr. OpenMPI ili sopstveno rješenje)
4. GPGPU programiranje (npr. OpenCL ili CUDA)

(10) Kombinovati dva navedena pristupa u cilju još većeg ubrzanja i dokumentovati rezultate. Pri tome je potrebno zabilježiti i grafički predstaviti rezultate prije i poslije primjene optimizacija i paralelizacije.

Obezbijediti nekoliko primjera ili jediničnih testova kojima se demonstriraju funkcionalnosti iz stavki u specifikaciji projektnog zadatka. Na pravilan način pokazati da će optimizovane varijante algoritma proizvoditi tačan rezultat.

Pridržavati se:

- principa pisanja čistog koda i pravilnog imenovanja varijabli, funkcija, klasa i metoda
- konvencija za korišteni programski jezik