# Multiprocesorski sistemi (SI4MPS)
# Drugi kolokvijum - popravni, 28.01.2009. godine

Literatura nije dozvoljena.
Kolokvijum traje 90 minuta.

1. Objasniti prednosti i nedostatke zajedničke keš memorije. [10 poena]

2. Uporedno razmotriti prednosti i nedostatke strategija invalidacije i ažuriranja. [10 poena]

3. Definisati model sekvencijalne konzsitencije i ograničen ja koja nameće. [10 poena]

4. Za protokol *Dragon* precizno objasniti: a) stanja, b) transakcije na magistrali i c) akcije protokla. Nacrtati i objasniti dijagram stanja i prelaza. [30 poena]

5. Dat je multiprocesorski sistem sa 4 identična procesora, koji koristi MSI za održavanje koherencije keš memorije. Svaka keš memorija ima po 2 ulaza, koji su veličine jedne reči. Preslikavanje je asocijativno, algoritam zamene je FIFO. Na početku su sve keš memorije prazne. Data je sledeća sekvenca pristupa memoriji:

| | | | |
|---|---|---|---|
| 1. P0,R,A0 | 4. P2,R,A0 | 7. P0,W,A0 | 10. P3,R,A2 |
| 2. P1,R,A1 | 5. P2,R,A0 | 8. P3,W,A0 | 11. P3,R,A0 |
| 3. P3,R,A1 | 6. P2,W,A0 | 9. P3,R,A0 | 12. P3,R,A2 |

   5.1. Koliko puta koji od procesora pristupa memoriji? [4 poena]
   5.2. Koliki je Hit Rate za svaki od procesora (brojati i čitanje i upis, prikazati zbirno)? [4 poena]
   5.3. Napisati stanja koherencije u svim procesorima i stanje memorije nakon svakog koraka (dovoljno je stanja ispisivati samo posle promene). [12 poena]

6. Napisati program na programskom jeziku C ili C++ koji računa broj π po Lajbnicovoj formuli. Obradu paralelizovati i ostvariti korišćenjem MPI. Svaki proces treba da obradi po 50 vrednosti n (najveća obrađena vrednost n će biti za 1 manja od broja procesa u MPI svetu pomnoženog sa 50). Proces sa rangom 0 treba da prikupi međurezultate od svih procesa, izračuna konačni rezultat i ispiše ga na standardnom izlazu. Prilikom obrade međurezultata, voditi računa o očuvanju maksimalne količine tačnosti konačnog rezultata. [20 poena]

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$

**Napomena:**
U zadacima pretpostaviti da funkcije koje obavljaju potrebne ulazne i izlazne radnje već postoje, tako da za njih samo treba navesti prototipove i pozvati ih na odgovarajućim mestima u programskom kodu. Pretpostaviti da korisnik unosi sintaksno ispravne podatke.
Ukoliko u bilo kom pitanju ili zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi bila lakše prepoznata prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke.