

---

Elektrotehnički fakultet u Beogradu  
Katedra za računarsku tehniku i informatiku

*Predmet:* Multiprocesorski sistemi (13E114MUPS)  
*Nastavnik:* dr Milo Tomašević, red. prof.  
*Asistent:* doc. dr Marko Mišić; Pavle Divović, dipl. ing.  
*Ispitni rok:* Avgust 2022.  
*Datum:* 24.08.2022.

*Kandidat\*:* \_\_\_\_\_

*Broj Indeksa\*:* \_\_\_\_\_

*Ispit traje 180 minuta, prvih sat vremena nije dozvoljeno napuštanje ispita.  
Upotreba literature nije dozvoljena.*

<i>Zadatak 1</i>	_____ /5	<i>Zadatak 6</i>	_____ /10
<i>Zadatak 2</i>	_____ /10	<i>Zadatak 7</i>	_____ /10
<i>Zadatak 3</i>	_____ /10	<i>Zadatak 8</i>	_____ /10
<i>Zadatak 4</i>	_____ /10	<i>Zadatak 9</i>	_____ /15
<i>Zadatak 5</i>	_____ /10	<i>Zadatak 10</i>	_____ /10

**Ukupno na ispitu:** \_\_\_\_\_ /100

**Napomena:** Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

\* popunjava student.

---

1. [5] Objasniti osobine eksplicitnih i implicitnih paralelnih programskih modela.

2. [10] Formalno definisati koherentan memorijski sistem. Koja je osnovna implikacija?

3. [10] Neka se dva procesa sinhronizuju preko zajedničkog flega  $x$ . Proces A čeka da  $x$  bude 0, a onda radi i zatim postavi  $x$  na 1. Proces B čeka da  $x$  bude 1, a onda radi i zatim postavi  $x$  na 0. Opisati neophodne transakcije u slučaju invalidacionog i ažurirajućeg protokola i izvesti zaključak.

4. [10] Opisati fenomene pravog i lažnog deljenja. Objasniti tehnike za smanjivanje ili eliminaciju lažnog deljenja.

5. [10] Objasniti kako studija o načinima deljenja podataka u aplikacijama utiče na strukturu kataloga u *directory* protokolima? U skladu sa tim, kakva je skalabilnost ovih protokola?

6. [10] Objasniti šta podrazumeva princip inkluzije u keš hijerarhiji. Objasniti šta je neophodno na implementacionom nivou da bi se održala inkluzija? Koje su prednosti, a koji *overhead*-i pri održavanju inkluzije?

7. [10] Objasniti gde i kada se završavaju poslovi generisani *task* direktivom kod OpenMP tehnologije. Da li programer može imati uticaja na to?

8. [10] Neka se posmatra isečak koda u prilogu napisan putem CUDA tehnologije za izvršavanje na grafičkom procesoru. Objasniti koji problem postoji u kodu i navesti zašto je on rešen korišćenjem atomskih operacija. Da li se rešenjem umanjuju performanse koda? Da li je alternativa korišćenje redukcije na nivou bloka?

```
__global__ void find_matches_kernel (
int* d_a, int d_an, int* d_b, int d_bn, int min_len,
int* d_matches, int* d_matches_ind) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    int idy = blockIdx.y * blockDim.y + threadIdx.y;
    int sub_len = 0;
    if (d_a[idx] == d_b[idy]) {
        while ((idx + sub_len < d_an) &&
                (idy + sub_len < d_bn) &&
                (d_a[idx + sub_len] == d_b[idy + sub_len]))
            sub_len ++;
        if (sub_len >= min_len) {
            d_matches [atomicAdd(d_matches_ind, 1)] =
                store_res(idx, idy, sub_len);
        }
    }
    return;
}
```

9. [15] Korišćenjem MPI tehnologije, paralelizovati kod u prilogu koji vrši izračunavanje srednje brzine niza čestica prilikom rešavanja nekog problema molekularne dinamike u 3D prostoru. Podaci brzini jedne čestice su dati u tri uzastopne lokacije nizu  $vh$ . Obratiti pažnju na efikasnost i korektnost paralelizacije.

```
double velavg(int npart, double vh[], double vaver, double h){
    int i;
    double vaverh = vaver * h, vel = 0.0, sq, count;
    count = 0.0;
    for (i = 0; i < npart * 3; i += 3){
        sq = sqrt(vh[i] * vh[i] + vh[i+1] * vh[i+1] + vh[i+2] *vh[i+2]);
        if (sq > vaverh)
            count++;
        vel += sq;
    }
    vel /= h;
    return(vel);
}
```

10. [10] Dat je multiprocesorski sistem sa 4 identična procesora, koji koristi *MESI* protokol za održavanje koherencije keš memorije. Svaka keš memorija ima po 2 ulaza, koji su veličine jedne reči. Preslikavanje je **direktno**. Početne vrednosti podataka su 0. Svaki upis uvećava vrednost izmenjenog podatka za 1. Na početku su sve keš memorije prazne. Data je sledeća sekvenca pristupa memoriji:

1. P2,R,A0	3. P0,W,A2	5. P1,R,A2	7. P1,R,A1
2. P2,W,A0	4. P2,R,A2	6. P1,W,A1	8. P0,W,A2

Napisati stanja koherencije u svim procesorima i stanje memorije posle svake promene i skicirati opisani sistem u trenutku 8. [8 poena]

Da li procesori pristupaju memoriji i kada? Za svaki pristup navesti razlog. [2 poena]

Trenutak 1											
P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 2											
P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 3											
P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 4											
P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 5

P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 6

P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 7

P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---

Trenutak 8

P0			P1			P2			P3		

Memorija	
A0	
A1	
A2	
A3	

Pristupi memoriji:

---