Dežurni nastavnik: _____

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Multiprocesorski sistemi
(13E114MUPS, SI4MPS, IR4MPS, MS1MPS)
Nastavnik: dr Milo Tomašević, vanr. prof.
Asistent: dipl. ing. Marko Mišić
Ispitni rok: jul 2015.
Datum: 02.07.2015.

Kandidat[*]*:* _____

Broj Indeksa[*]*:* _____

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

Zadatak 1	_____/5	*Zadatak 6*	_____/5
Zadatak 2	_____/10	*Zadatak 7*	_____/15
Zadatak 3	_____/15	*Zadatak 8*	_____/10
Zadatak 4	_____/10	*Zadatak 9*	_____/10
Zadatak 5	_____/10	*Zadatak 10*	_____/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] Navesti i objasniti glavne namene paralelnih računara.

2. [10] Objasniti šta je paralelni programski model i nabrojati osnovne paralelne programske modele? Na koje projektne odluke i kako utiče izbor paralelnog programskog modela?

3. [15] Kod protokola MSI precizno:
   a) nacrtati dijagram stanja i detaljno opisati sve akcije
   b) dati dokaz koherencije.

4. [10] Šta odlučujuće utiče na izbor strategije koherencije kod *snoopy* protokola? U tom smislu diskutovati performanse ažurirajućih i invalidacionih protokola.

5. [10] Objasniti kako izgleda katalog kod Dir_i *SW* protokol. Objasniti odvijanje tipičnih akcija i osnovne osobine ovog protokola.

6. [5] Koje se prednosti dobijaju ako se u jednom čvoru hijerarhijskog sistema nalazi više procesora?

7. [15] Korišćenjem OpenMP tehnologije, paralelizovati funkciju koja je data u prilogu. Obratiti pažnju na efikasnost paralelizacije.

```
void process (int cols, int penalty, int *in, int *ref) {
  int idx, index;
  for( int i = 0 ; i < cols-2 ; i++){
        for( idx = 0 ; idx <= i ; idx++){
         index = (idx + 1) * cols + (i + 1 - idx);
         in[index]= max(in[index-1-cols] + ref[index],
                        in[index-1] - penalty, in[index-cols] - penalty);
        }
  }
  for( int i = cols - 4 ; i >= 0 ; i--){
    for( idx = 0 ; idx <= i ; idx++){
          index = (cols - idx - 2) * cols + idx + cols - i - 2;
          in[index]= max(in[index-1-cols]+ ref[index],
                         in[index-1] - penalty, in[index-cols] - penalty);
          }
  }
}
```

Rešenje:

```
void process (int cols, int penalty, int *in, int *ref) {
  int idx, index;


  for( int i = 0 ; i < cols-2 ; i++){


        for( idx = 0 ; idx <= i ; idx++){


         index = (idx + 1) * cols + (i + 1 - idx);


         in[index]= max(in[index-1-cols] + ref[index],
                        in[index-1] - penalty, in[index-cols] - penalty);

        }
  }


  for( int i = cols - 4 ; i >= 0 ; i--){


    for( idx = 0 ; idx <= i ; idx++){


          index = (cols - idx - 2) * cols + idx + cols - i - 2;


          in[index]= max(in[index-1-cols]+ ref[index],
                         in[index-1] - penalty, in[index-cols] - penalty);
    }
  }
}
```

8. [10] U prilogu je dato jezgro napisano na CUDA tehnologiji koje vrši operaciju redukcije. Da li je priloženo jezgro ispravno? Ukoliko nije, dopisati i objasniti neophodne izmene, tako da jezgro ispravno vrši operaciju redukcije. Smatrati da jedan blok sadrži 256 niti.

```
__global__ void reductionSum (int* devA, int* blockResults, int n) {
  extern __shared__ int sharedData[];
  unsigned int tid = threadIdx.x;
  unsigned int i = blockIdx.x * blockDim.x + threadIdx.x;
  if (i < n) sharedData[tid] = devA[i]; else sharedData[tid] = 0;
  __syncthreads();
  if (tid < 128) sharedData[tid] += sharedData[tid + 128];
  if (tid < 64) sharedData[tid] += sharedData[tid + 64];
  if (tid < 32) sharedData[tid] += sharedData[tid + 32];
  if (tid < 16) sharedData[tid] += sharedData[tid + 16];
  if (tid < 8) sharedData[tid] += sharedData[tid + 8];
  if (tid < 4) sharedData[tid] += sharedData[tid + 4];
  if (tid < 2) sharedData[tid] += sharedData[tid + 2];
  if (tid < 1) sharedData[tid] += sharedData[tid + 1];
  if (tid == 0) blockResults[blockIdx.x] = sharedData[0];
}
```

9. [10] Šta rade i čemu služe MPI operacije *Scatter* i *Gather*? Da li su ove operacije blokirajuće? Navesti primer korišćenja *Scatter* operacije u MPI svetu sa 4 procesa kod koga se u procesu sa rangom 0 nalazi niz od 100 elemenata.

10. [10] Dat je multiprocesorski sistem sa 4 identična procesora, koji koristi *Firefly* 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. P0,R,A0 2. P1,R,A0	3. P1,W,A0 4. P2,R,A2	5. P2,W,A2 6. P0,R,A2	7. P2,W,A0 8. P1,R,A1

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:

_____