(Svako pitanje donosi 3 boda za točan odgovor. Zadaci na drugoj stranici su možda nešto teži – preporuka najprije riješiti zadatke s prve stranice.)

Rješenja su zadana samo za zadatke 6-10.

1. Čemu služi operacijski sustav?
   Koji se registar sprema na stog pri pozivu potprograma?
   Kojom operacijom se obavlja instrukcija skoka na adresu X (što radi instrukcija skoka)?

2. Neki sustav koji koristi sklop za prihvat prekida prima zahtjeve za prekid: u t=1 ms dolazi P2, u t=4 ms dolazi P3 te u t=6 ms dolazi P1. P3 ima najveći prioritet, slijedi P2 te P1. Obrade prekida traju po 3 ms. Prihvat prekida i povratak iz prekida traju po 0,5 ms. Grafički prikazati rad procesora za vrijeme od t=0 ms do t=15 ms sa stanjima: GP, P1, P2, P3, PP, PiP.

3. Jedan posao predstavljen je lancem podzadataka Z1→Z2→...→Z8. Za podzadatke su poznate domene i kodomene prema tablici (stupci su podzadaci, redci su zajedničke strukture podataka):

M\Z	1	2	3	4	5	6	7	8
1	D	–	–	D	–	–	K	–
2	K	–	D	K	–	–	–	D
3	–	D	K	–	D	–	D	K
4	–	K	–	–	K	D	–	–
5	–	–	–	D	–	K	–	D

   Odrediti maksimalno paralelan sustav podzadataka i prikazati ga usmjerenim grafom.

4. Navesti jezgrine funkcije jednostavna modela jezgre korištene na predavanjima. Uz svaku funkciju navesti tko ju poziva (dretva ili naprava) te može li poziv blokirati pozivajuću dretvu (smjestiti u red različiti od `Aktivna_D` i `Pripravne_D`) ili odblokirati neku drugu dretvu.

5. Stanje sustava u nekom trenutku definirano je podacima jezgre:

```
Aktivna_D = 4
Pripravne_D = 3 2 6
Odgođene_D = 5(1) 7(4)
UI[1] = 1
OSEM[1] = - (prazno)
OSEM[1].v = 2
```

   Pozivi jezgre koje izvode aktivne dretve ili se pozivaju iz obrade prekida su redom:

```
1. PostaviOSEM(1)
2. Zakasni(3)
3. ČekajOSEM(1)
4. Prekid_UI(1)
```

   Pokazati stanje sustava nakon svakog gornjeg poziva jezgre. Zbog sitne razlike u ostvarenju jezgrinih funkcija navesti koriste li se one definirane u knjizi ili one iz skripte (rješenje nije isto).

6. U nekom proizvodnom sustavu upravljanje izradom nekog proizvoda ostvareno je s tri cikličke dretve. Prve dvije upravljaju sastavljanjem proizvoda od elemenata koji se nalaze na traci dok treća pomiče traku kad obje prethodne dretve završe radom. Potom se postupak ponavlja (prve dvije sklapaju idući proizvod ...). Neka se operacije prvih dviju dretvi opišu sa pozivima `sklopi_1()` i `sklopi_2()` a treće s `pomakni()`. Napisati pseudokod za sve tri dretve koristeći se navedenim funkcijama za same operacije te binarnim i/ili općim semaforima za sinkronizaciju. Navesti početne vrijednosti semafora ukoliko je početno stanje takvo da se prve dvije dretve mogu odmah pokrenuti sa svojim operacijama.

```
dretva sastavi_1(){
   ponavljaj {
      sklopi_1()
      PostaviOSEM(1)
      ČekajOSEM(2)
   }
   do zauvijek
}
```

```
dretva sastavi_2(){
   ponavljaj {
      sklopi_2()
      PostaviOSEM(1)
      ČekajOSEM(3)
   }
   do zauvijek
}
```

```
dretva pomak(){
   ponavljaj {
      ČekajOSEM(1)
      ČekajOSEM(1)
      pomakni()
      PostaviOSEM(2)
      PostaviOSEM(3)
   }
   do zauvijek
}
```

```
Početne vrijednosti semafora su nule.
Za potpuno točno rješenje trebaju 3 semafora!
```

7. *N* jednakih dretvi surađuju na izvođenju ista posla koji se sastoji od *M* iteracija. Kada jedna od dretvi završi posao u nekoj iteraciji, ona mora pričekati da to naprave i sve ostale prije nego li će krenuti s idućom iteracijom. Ukoliko se s `posao()` označi posao dretve unutar jedne iteracije napisati pseudokod za dretve (isti kod za sve dretve). Koristiti semafore (i po potrebi dodatne varijable). Pri pokretanju, dretve mogu odmah krenuti s prvom iteracijom.

```
dretva {
   za i=1 do M {
      posao()
      ČekajBSEM(1)
      br++
      ako je ( br < N ) {
         PostaviBSEM(1)
         ČekajOSEM(2)
      }
      inače { //zadnja dretva resetira i pokreće iduću iteraciju
         br = 0
         PostaviBSEM(1)
         za j=1 do N-1
           PostaviOSEM(2)
      }
   }
}
zajednička (globalna varijabla): br=0
semafori:
BSEM[1].v = 1
OSEM[2].v = 0
```

8. Skup dretvi obavlja slijedeći kod (`i`=jedinstveni identifikator dretvi, iz skupa {`1..N`}:

```
dretva(i) {
   za j = 0 do M radi
      k = i + j*N
      ako je k > M
         završi_s_radom();
      za m = 1 do M radi
         c[k,m] = 0
         za n = 1 do M radi
            c[k,m] + = a[k,n] * b[n,m]
}
```

Jesu li dretve međusobno zavisne ili ne (treba li ih sinkronizirati ili ne)? Obrazložiti.

Dretve su nezavisne. Svaka dretva „i“ izračunava retke: i, i+N, i+2N, … izlazne matrice (c=a*b).

U zadatku je možda trebalo pisati da je varijabla k lokalna za pojedinu dretvu, a matrice a[], b[] i c[] zajedničke (za j, m i n je očito da su lokalne varijable). Stoga će ocjenjivanje ovog zadatka uključiti i obrazloženja „zbog 'k' su dretve zavisne“ kao ispravna.

9. Zadan je slijedeći algoritam s namjenom korištenja za međusobno isključivanje:

```
//zajedničke varijable
BROJ[N]= {0}; //0 je početna vrijednost svim brojevima
NA_REDU = 1;

uđi_u_KO(i) {
   BROJ[i] = NA_REDU;
   za k = 1 do N radi
      ako je ( k <> i ) I ( BROJ[k] >= BROJ[i] ) tada
         BROJ[i] = BROJ[k]+1;

   dok je NA_REDU < BROJ[i] radi //radno čekanje
         ;
}
izađi_iz_KO(i) {
   NA_REDU = NA_REDU + 1;
}
```

Koje uvjete prema algoritmima međusobnog isključivanja gornji algoritam ne zadovoljava? Obrazložiti na primjeru/ima izvođenja koji dovodi do problema.

Može se dogoditi potpuni zastoj: ako razmatramo sustav od samo dvije dretve i te dretve paralelno uzimaju broj (`BROJ[i] = NA_REDU`) tada će u idućoj petlji svaka povećati svoj broj barem za jedan te niti jedna neće izaći iz iduće petlje (`dok je NA_REDU < BROJ[i] radi`) jer obje imaju broj veći od `NA_REDU`.

10. Neka naprava stvara novi podatak svakih 100 mikrosekundi. U međuspremniku same naprave stane 100 takvih podataka. Napravom se može upravljati a) radnim čekanjem; b) prekidima na svaki primljeni podatak; c) prekidima na događaj popunjenja internog međuspremnika naprave; d) sklopom s izravnim pristupom spremniku (DMA). Prihvat prekida i povratak iz prekida (ne uključujući *obradu*) traju 100 sabirničkih ciklusa ukupno (sabirnički ciklus traje 10 ns). Programski prijenos jednog podatka u radni spremnik zahtjeva dva sabirnička ciklusa. DMA sklop izaziva programski prekid nakon što prenese 1000 podataka (sama *obrada* tog prekida traje 10 sabirničkih ciklusa).
i) Koliko procesorskog vremena preostaje za druge potrebe ukoliko se koristi a, b, c ili d način upravljanja napravom?
ii) (za dodatna 3 boda, izvan „kvote“ kolokvija) Uz pretpostavku da u sustavu postoji dretva koja može obraditi prispjele podatke čim su oni spremljeni u radni spremnik, koliko u prosjeku svaki podatak čeka na početak obrade (za a-d)?

i)
a) ostaje 0% jer se svo „slobodno“ vrijeme troši na radno čekanje
b) unutar 100 us (us = mikrosekunda): 100+2 sabirnička ciklusa = 102*10 ns = 1,02 us => ostaje 100-1,02 = 98,98 us što je ujedno i postotak (98,98 us / 100 us = 98,98 %)
c) prekid nakon 100 novih podataka, tj. nakon 100*100 us;
obrada prekida sada traje: 100 + 100*2 = 300 sabirničkih ciklusa = 300 * 10 ns = 3 us
ostaje: 100*100 us – 3 us = 9997 us => 9997/10000 = 99,97 %
d) 1000 podataka = 1000 * 100 us
potrebno sabirničkih ciklusa: 1000*1 + 100+10 = 1110 => 1110 * 10 ns = 11,1 us
ostaje: 1000*100 us – 11,1 us = 99988,9 us => 99988,9/100000 = 99,9889 %

ii)
Pretpostavka je da imamo višeprocesorski sustav i da ta druga dretva može paralelno raditi s dohvatom podataka (na bilo koji način).
Inače:
a) ne bi imalo smisla;
b) i c) trebalo bi obračunati cijeli postupak prihvata prekida (a ne samo pola kao u donjem rješenju), tj. umjesto 50 staviti 100
d) nema utjecaja jer dretva već u idućem sabirničkom ciklusu može dohvatiti podatak

a) radno čekanje koristi programski prijenos podatka u radni spremnik, a to znači da je podatak u radnom spremniku za dva ciklusa => tolika je i odgoda: 2*10 ns = 20 ns
(mogli bi na to dodati i još par sabirničkih ciklusa: dohvat registra PR, provjera ZASTAVICE, …)

b) nakon prihvata prekida i prijenosa podataka: 100/2 + 2 = 52 ciklusa => 520 ns

c) gledamo 100 podataka (jer se nakon toga sve opet ponavlja)
prvi podatak mora čekati da dođe još 99; 100-ti će tek izazvati prekid
vrijeme do prijenosa u spremnik za i-ti podatak = (100-i)*100 us + (50 + (i-1)*2 + 2) * 10 ns =
= (100-i)*100 us + (50+i*2) ns = 10000 – 100*i + 0,05 + 0,002*i us = 10000,05 – 99,998 * i us
u prosjeku: 10000,05 – 99,998 * (100*101/2 / 100) = 10000,05 – 99,998 * 50,5 = 4950,101 us

d) DMA sklop podatak prenese u 1 sab. ciklusu, te je odgoda 10 ns