Operacijski sustavi – prvi kolokvij, 10. travnja 2013.

Sve zadatke osim 6. i 12. rješavati na ovom papiru

_____

Ime Prezime

1. U slijedećem programu koje funkcije koriste OS (podcrtati ih)? Koja funkcija koristi koji podsustav (napisati sa strane)?

```
int main () {
   int i, j=0;
   FILE *fp = fopen ( "suma1.txt", "w" );        datotečni podsustav
   for ( i = 0; i < N ; i++ ) {
      j += sqrt (i);
      printf ( "Iteracija: %d %d\n", i, j );     UI podsustav
      fprintf ( fp, "%d %d\n", i, j );           datotečni podsustav
   }
   fclose ( fp );                                datotečni podsustav
   return 0;
}
```

2. U jednostavnom sustavu u kojem se sve nalazi u radnom spremniku (procesor nema priručni spremnik) izvode se niže navedene instrukcije. Koliko sabirničkih ciklusa je potrebno za izvođenje navedenih instrukcija (navesti zasebno za svaku instrukciju), ne računajući dohvat same instrukcije?

```
LDR R0, (R4)        1 ciklus = učitavanje sadržaja s adrese sadržane u R4 u R1

STR R1, (R5)        1 ciklus = spremanje sadržaja iz R0 na adresu iz R4

CALL funkcija1      1 ciklus = spremanje PC na stog

RET                 1 ciklus = učitavanje PC sa stoga
```

3. Što je to kontekst dretve?

   sadržaj svih registara procesora

4. Opisati načine komunikacije s ulazno-izlaznim napravama.

   radno čekanje: programski se čeka da naprava bude spremna za komunikaciju; kada bude (provjera preko zastavice u registru stanja) čitaju/pišu se podaci

   prekidni način: naprava signalizira spremnost za komunikaciju prekidom; u obradi prekida se dohvaćaju podaci s naprave ili na nju pokranjuju drugi (ili naredbe)

5. Neka ulazna naprava koristi se mehanizmom radnog čekanja. Ako naprava prosječno generira podatak svake 1 ms koliko će se ukupno sabirničkih ciklusa utrošiti na obradu jednog podatka, ako je frekvencija sabirnice 100 MHz?

   100 MHz * 1 ms = 100 000 000 1/s * 0,001 s = 100 000 sabirničkih ciklusa

6. U sustavu bez sklopa za prihvat prekida, kod kojeg se prekidi obrađuju redom prispijeća uz zabranjeno prekidanja za vrijeme obrada, javljaju se slijedeći zahtjevi za prekid: (vrijeme, oznaka_zahtjeva) = { (1, P1), (3, P2), (7, P3) }. Ukoliko postupak prihvata prekida (PPP) traje 1 ms, postupak povratka iz prekida (PIP) 0,5 ms te obrade prekida po 2 ms, grafički prikazati stanje procesora pri izvođenju tih prekida. Koliko je procesorskog vremena utrošeno na račun tih prekida?

   (jednostavno i uglavnom dobro rješeno)

7. Navesti uvjet nezavisnosti dvije dretve: $(D_i \cap K_j) \cup (K_i \cap D_j) \cup (K_i \cap K_j) = \emptyset$

Utvrditi jesu li slijedeće dvije dretve zavisne ili ne (sve korištene varijable su globalne=zajedničke). Obrazložiti odgovor.

```
dretva1 {
   za i = 1 do N radi {
      A[i] += B[i] + C[i];
      B[i] += A[i];
   }
}
```

```
dretva2 {
   za j = N+1 do M radi {
      A[j] += B[j] + C[j];
      B[j] += A[j];
   }
}
```

Dretve su nezavisne jer rade na različitim skupovima podataka

8. Navesti zajednički nedostatak slijedećih algoritama međusobnog isključivanja: Dekkerov algoritam, Lamportov algoritam, isključivanje instrukcijom TAS.

radno čekanje

9. Zadan je slijedeći algoritam:

```
uđi_u_KO(I) {
   dok je ZASTAVICA != 0 radi
      ;
   ZASTAVICA = I;
}
```

```
izađi_iz_KO(I) {
   ZASTAVICA = 0;
}
```

Početna vrijednost `ZASTAVICA=0`.
Zašto algoritam nije dobar za međusobno isključivanje dretvi? Obrazložiti.

Više dretvi može ući u KO ako u uzastopnim sabirničkim ciklusima čitaju vrijednost varijable `ZASTAVICA`

10. Navesti sve strukture podataka jezgre za jednostavan model jezgre prikazan na predavanjima.

Opisnici UI naprava: UI[]
Opisnici dretvi: za svaku dretvu opisnik s elementima: id, prioritet, Zadano_kašnjenje, kontekst
Opisnici listi za stanje dretvi: Aktivna_D, Pripravne_D, Odgođene_D (+ Postojeće_D)
Opisnici semafora: Sem[] s elementima: vrijednost semafora, red za blokirane dretve

11. Kako se ulazi u jezgru (kako se pozivaju jezgrine funkcije, kojim mehanizmom)?

mehanizmom prekida

12. Korištenjem semafora sinkronizirati cikličke dretve *vozač* i *radnik*. Dretva *vozača* simulira vozača koji prevozi radnika na jednoj kružnoj ruti. Svaki puta kada vozač stane, radnik izađe i napravi svoj posao na toj postaji (npr. pokupi smeće iz koša). Akcije koje vozač radi u petlji simulirati sa: *otvori()*, *zatvori()*, *vozi()* (tim redom). Akcije koje radnik radi simulirati sa: *uđi()*, *izađi()*, *radi()* (tim redom). Sinkronizirati samo neophodne radnje: ulazak u vozilo te izlazak iz vozila (vrata u oba slučaja moraju biti otvorena, a zatvaraju se po ulasku radnika).
(okreni)

```
dretva vozač()
{
   ponavljaj {
      PostaviSem(OTVORENO);
      ČekajSem(UNUTRA);
      zatvori();
      vozi();
      otvori();
   }
}
```

```
dretva radnik()
{
   ponavljaj {
      ČekajSem(OTVORENO);
      izađi();
      radi();
      uđi()
      PostaviSem(UNUTRA);
   }
}
```

Početno stanje: vozač i radnik u automobilu, vrata zatvorena
Poč. vr. sem. = 0 za oba.

Kada bi početno stanje bilo da je radnik vani, onda bi u prikazani kod vozača prije petlje trebalo dodati:

```
otvori();
PostaviSem(OTVORENO);
ČekajSem(UNUTRA);
```

a u kod radnika prije petlje trebalo bi dodati

```
ČekajSem(OTVORENO);
uđi()
PostaviSem(UNUTRA);
```