1. UVOD

1. Što je Operacijski sustav?

Skup programa koji olakšavaju rad na računalu, tj. skup programa koji omogućuju provođenje raznih zahvata na računalu.

2. Navesti osnovne dijelove operacijskog sustava.

Mrežni podsustav, sigurnosni podsustav, U/I podsustav, upravljanje memorijom, datotečni podsustav, procesi, dretve, sinkronizacija, raspoređivač poslova, komunikacija, API, GUI.

3. Program čiji je izvorni kod u datoteci lab5.c, kompajlira se sa: ***lab5.c***

a pokreće sa ***./a.out*** .

4. Ako program iznenada završi s porukom "Segmentation Fault" što treba razmatrati pri ispravljanju greške?

5. Što je to sučelje? Što je to API (tko ga nudi, tko koristi)?

Sučelje je način komuniciranja kod kojeg način postavljanja zahtjeva operacijskom sustavu, kao i izgled povratnih poruka operacijskog sustava mora biti dogovoren.

API (*Application Programming Interface*) su funkcije pripremljene unutar operacijskog sustava koje su dohvatljive sučelju prema primjenskim programima. Koriste ga programeri primjenskih programa.

2. Model jednostavnog računala

6. Čime su određena svojstva i ponašanje procesora?

Skupom registara i skupom instrukcija.

7. Navesti osnovni skup registara procesora.

Adresni, podatkovni, registar stanja, PC, SP, instrukcijski registar i registar opće namjene.

8. Što je to sabirnički ciklus?

Vrijeme za koje sabirnica u jednom sabirničkom ciklusu trajanja T_B ostvaruje jednu vezu, tj. prijenos jednog sadržaja.

9. U pseudokodu napisati što procesor trajno radi

```
ponavljati {
      dohvati instrukciju na koju pokazuje PC;
      dekodirati instrukciju;
      povećati PC;
      odrediti izvor operanada i destinaciju rezultata;
      operande dovesti na ALU;
      izvesti operaciju;
      pohraniti rezultat;
} dok je(procesor uključen);
```

10. Što je kontekst dretve?

Sadržaj registara procesora kojeg treba pri prekidanju izvođenja jedne dretve pohraniti izvan procesora i u registre staviti sadržaj dretve koju se želi pokrenuti.

11. Što se zbiva pri izvođenju instrukcije za poziv potprograma?

```
Ponavljati {
            Iz spremnika dohvatiti instrukciju na koju pokazuje
            programsko brojilo;
            Dekodirati instrukciju, odrediti operaciju koja se treba
            izvesti;
            Povecati programsko brojilo, da pokazuje na sljedecu
            instrukciju;
            Ako je (dekodirana instrukcija poziv potprograma) {
                        Pohraniti sadržaj programskog brojila na stog;
                        Smanjiti sadržaj registra SP,tako da pokazuje
                        na sljedece prazno mjesto;
                        Iz adresnog dijela instrukcije odrediti adresu
                        pocetka potprograma;
                        Staviti adresu u programsko brojilo;
            }
            Inače
            Obaviti instrukciju na način određen dekodiranim
            operacijskim kodom;
      }
      Dok je (procesor ukljucen);
```

12. Definirati osnovno pojmove: program, proces, dretva.

Program je statični niz instrukcija pohranjen na papiru, memoriji ili disku.

Dretva je niz instrukcija u izvođenju koje kontroliraju proces.

Program koji se smjesti u radni spremnik i pokrene pritom dobivajući dinamička obilježja.

13. Kako je moguć višeprogramski rad na jednoprocesorskom računalu?

Tako da procesor naizmjence izvodi sve dretve.

3. Obavljanje UI operacija, prekidni rad

14. Skicirati način spajanja UI naprave na sabirnicu.

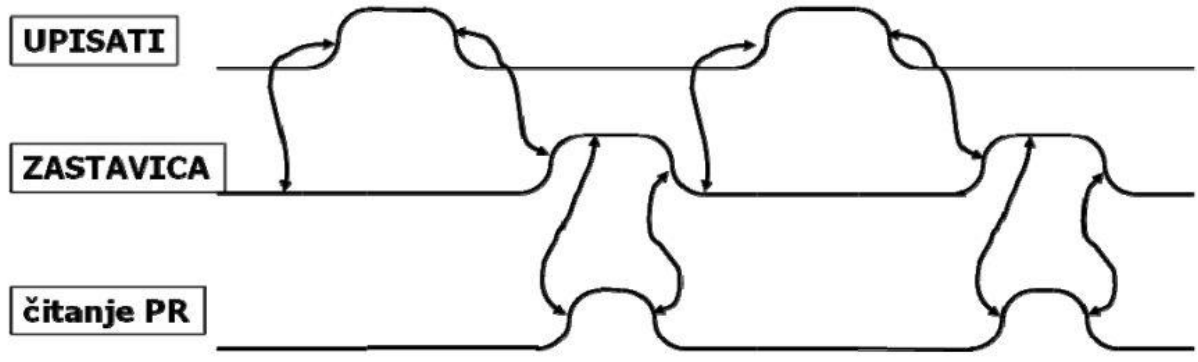
rucno

15. Što je radno čekanje?

Neprestano ispitivanje uvjeta.

```
dok je (zastavica==0)
      čitaj PR;
```

16. Skicirati signale dvožičnog rukovanja.



17. Što se zbiva kada se dogodi prekid?

Procesor prelazi u sustavski način rada te se poziva odgovarajuća jezgrina funkcija.

18. Kako treba nodopuniti ponašanje procesora da on omogućuje prekidni rad bez sklopa za prihvat prekida?

```
Pojavio se prekidni signal, zabranjeno je prekidanje, i programsko brojilo nalazi se na sustavskom stogu;
     {
          Pohrani kontekst;
          Ustanovi uzrok prekida, odnosno odredi indeks prekida i;
          Ako je (1<i<n){
               Postavi oznaku cekanja K_Z[i]=1;
               Ponisti zastavicu u registru stanja prekida i;
               Dok je ((postoji K_Z[j]) != 0) ?(J>T_P)){
                    Odabrati najveci j;
                    K_Z[j]=0;
                    Pohraniti kontekst sa sustavskog stoga i T_P u KON[j];
                    T_P=j;
                    Omoguci prekidanje;
                    Predji u korisnicki nacin rada;
                    Pozovi podprogram za obradu prekida j;
                    Zabrani prekidanje;
                    Predji u sustavski nacin rada;
                    Vrati na sustavski stog i u varijablu T_P sadrzaj iz KON[j];
               }
          }
          Obnovi kontekst sa sustavskog stoga;
          Omoguci prekidanje;
          Vrati se iz prekidnog nacina rada;
     }
```

19. Pojasniti instrukcije „pohraniti kontekst“ i „vratiti se iz prekidnog načina“?

Pohraniti kontekst znači pohraniti sadržaje svih registara procesora na sustavski stog.

Vratiti ti se iz prekidnog načina rada znači da se u PC vraća vrijednost programskog brojila sa stoga, aktivira se korisnička kazaljka stoga, adresira se korisnički dio spremnika.

20. Što treba načiniti na početku svakog podprograma za obradu prekida?

Treba pohraniti sadržaje svih registara procesora na sustavski stog.

21. Zašto se programsko brojilo tretira zasebno prilikom pohrane konteksta?

Zato da se procesor može vratiti na točno „ono“ mjesto na kojem je bio prekinut.

22. Što se zbiva kada obrada nekog prekida završi?

Obnavlja se kontekst sa sustavnog stoga, omogućava se prekidanje, vraća se iz prekidnog načina rada.

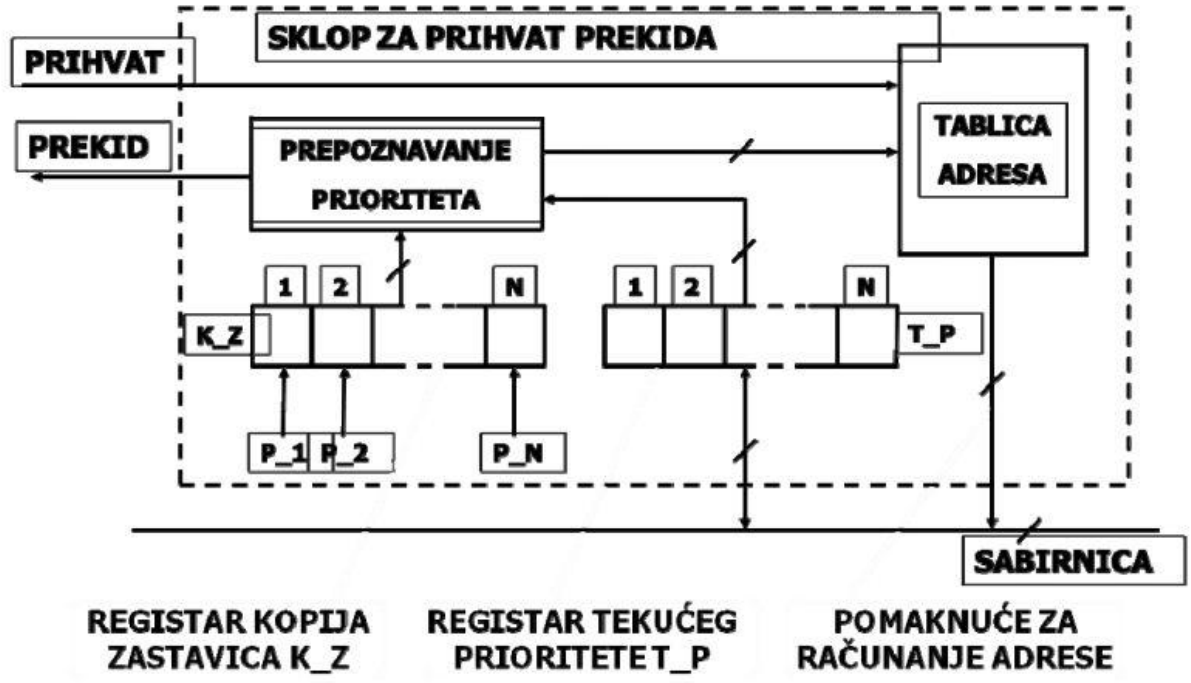
23. Koje strukture podataka treba sadržavati operacijski sustav koji omogućuje prihvat

prekida različitih prioriteta?

Varijablu T_P (tekući prioritet), polje KON[N] (kontekst pojedine dretve) i polje

K_Z[N] (kopije zastavica).

24. Opisati sklop za prihvat prekida.



25. Kako treba nodopuniti ponašanje procesora da on omogućuje prekidni rad sa sklopom za prihvat prekida?

```
Ako je (prekidni signal postavljen){
            Zabraniti prekidanje;
            Prebaciti adresiranje u sustavski adresni prostor i aktivirati sustavsku kazaljku stoga;
            Postaviti signal PRIHVAT;
            Pohraniti programsko brojilo i sve ostale registre na sustavski stog
            Sa sabirnice preuzeti sadrzaj i iz njega odrediti adresu prekidnog podprograma;
            Staviti tu adresu u programsko brojilo;
}
```

26. Navesti koje sve radnje mogu generirati prekide unutar procesora.

Dijeljenje s nulom, adresiranje nepostojeće lokacije u adresnom prostoru, dekodiranje nepostojeće instrukcije.

27. U kojem slučaju će se dogoditi „poziv jezgre“, odnosno „ulazak u jezgru“ i što se tada poziva?

Kada se dogodi prekid. Tada se poziva odgovarajuća jezgrina funckija.

28. Navesti osnovne registre prisrupnog sklopa za neposredni pristup spremniku (DMA).

Adresni registar, brojač podataka, podatkovni registar i registar stanja.

29. U pseudokodu napisati programski odsječak koji obavlja sklop za neposredni pristup spremniku.

```
dok je(BR>0){
      zatražiti sabirnicu; čekati sabirnicu;
      postaviti na adresni dio sadržaj iz AR;
      prenjeti na tu adresu sadržaj uz PR;
      AR++;
      BR--;
}postaviti signal PREKID;
```

30. Opisati čvrsto povezani višeprocesorski sustav.

Sustav se sastoji od N procesora od kojih svaki ima svoj lokalni spremnik u koi samo on moze pristupiti. Osim toga, svaki proceor može pristupiti do jednoh zajednickog djeljenog spremnika preko zajednicke sabirnice. U jednom sabirnickom ciklusu do spremnika moze pristupiti samo jedan od procesora. Da bi odredili koji procesor moze pristupiti spremniku imamo posebni sklopovski dodjeljivac sabirnice. Procesor I koji zeil pristup do djeljenog spremnika postavlja signal trazenja sabirnice T[I] dodjeljivacu sabirnice. Dodjeljivac na pocetku svog spremnickog ciklusa odlucuje kojem ce procesoru dodjeliti sabirnicu. U jednom spremnickom ciklusu dodjeljivac ce samo jednom procesoru dodjeliti sabirnicu, odnosno dodjeliti mu D[I]. Ako neki procesor postavi svoj zahtjev za dodjelu sabirnice on ce u svom izvodjenju zastati dok mu se sabirnica ne dodjeli. Dodjeljivac sabirnice dodjeljuje sabirnicu ciklicki.

4. Međusobno isključivanje u višedretvenim sustavima

31. Što je zajedničko procesu roditelju i procesu djetetu? Koje računalne resurse dijele dretve

istog procesa?

Ništa, dijete ima kopiju podataka i instrukcija roditelja, no ništa im nije zajedničko. Dretve istog procesa dijele sve računalne resurse.

32. Kako je podijeljen spremnički prostor procesa, a kako dretveni spremnički podprostor?

Spremišni prostor procesa u sebi ima N4 dredtvene podprostore i zajedničke (globalne) varijable, dretveni spremnički podprostor je podijeljen na tri dijela: instrukcija, stog i lokalni podaci.

33. Navesti uvjet nezavisnosti dretvi.

$(X \cap Y) \cup (X \cap Y) \cup (Y \cap Y) = O$ (prazan skup)

34. Navesti uvjete koje mora zadovoljavati algoritam međusobnog isključivanja dretvi.

Samo jedna dretva u datom trenutku smije biti u kritičnom odsječku.

Algoritam mora vrijediti i kad brzina izvršavanja dretvi nije jednaka.

Ako jedna dretva zastane u svom nekritičnom odsječku, ona ne smije time spriječavati drugu dretvu da uđe u svoj kritični odsječak.

Ako dvije dretve istodobno zatraže ulaz u kritični odsječak, odluka koja će dretva ući mora se napraviti u konačnom vremenu.

35. Za zadani algoritam međusobnog isključivanja ustanoviti je li ispravan. Obrazložiti odgovor.

```
Dretva I{
 dok je (1){
      dok je (ZASTAVICA[J] != 0);
            ZASTAVICA[I] = 1;
            kritični odsječak;
            ZASTAVICA[I] = 0;
       nekritični odsječak;
      }
}
```

36. Čemu služi Dekkerov, a čemu Lamportov algoritam? Koje strukture podataka koriste?

Dekkerov algoritam služi za međusobno isključivanje dvije dretve, a Lamportov algoritam služi za međusobno isključivanje N dretvi.

Deker koristi: varijablu PRAVO i ZASTAVICA, a Lamport koristi: polja BROJ[N] i ULAZ[N] te varijablu ZADNJI_BROJ.

37. Navesti Dekkerov/Lamportov algoritam.

Dekker

dok je (1) {

ZASTAVICA = 1;

čitaj ZASTAVICA[j];

dok je (ZASTAVICA[j] != 0) {

ako je (PRAVO != i) { ZASTAVICA[i] = 0; dok je (PRAVO != i) {

čitaj PRAVO;

} ZASTAVICA[i] = 1; } čitaj ZASTAVICA[j];

} K.O.;

pravo = j;

zastavica[i] = 0; N.K.O.

}

Lamport

Pseudokod:

```
dok je (1) {
    ULAZ[i] = 1;
    citaj ZADNJI BROJ;
    BROJ[i] = ZADNJI BROJ + 1;
    ZADNJI BROJ = BROJ[i];
    ULAZ [i] = 0;
}                                   } Ulaz kroz vrata

za(j=0; j<N; j++) {                 } Pregledavanje svih dretvi
        citaj ULAZ[j];
        dok je (ULAZ[j] == 1) {     } Pričekati ako dretva j upravo prolazi kroz vrata
                citaj ULAZ[j];
        }

        čitaj BROJ[j];              } Pregledavanje svih dretvi
        dok je ((BROJ[j]!=0 && (BROJ[j], j)<(BROJ[i],i)) {
                čitaj BROJ[j];
        }
}

K.O.
BROJ[i] = 0;

N.K.O
```

38. Usporediti Petersonov i Dekkerov algoritam.

Kod Petersona se za razliku od Dekkera pravo dodjeljuje na početku, jednostavnije je radno čekanje, ako je neka dretva brža prije ulazi u kritični odsječak.

39. Navesti najjednostavniji način međusobnog isključivanja više dretvi na jednoprocesorskom računalu.

Prekidima. Kada dretva zeli uci u K.O. ona zabrani prekide, i na izlasku iz K.O. ih ponovno omoguci.

40. Navesti nedjeljive instrukcije procesora koje služe kao sklopovska potpora međusobnom isključivanju.

TAS (ispitati i postaviti) - u prvom ciklusu dobave sadržaj adresirane lokacije i smjeste ga u jedan od registara procesora, a u drugom ciklusu pohranjuju u tu lokaciju vrijednost 1
swap (zamjeni)- u prvom ciklusu dobave sadržaj adresirane lokacije i smjeste ga u jedan od registara procesora, a u drugom ciklusu pohranjuju u tu lokaciju vrijednost koja je prije toga bila pohranjena u tom ili drugom registru
fetch-and-add - u prvom ciklusu dobave sadržaj adresirane lokacije i smjeste ga u jedan od registar procesora, a u drugom ciklusu pohranjuju na tu lokaciju taj sadržaj uvećan za jedan

41. U pseudokodu riješiti problem međusobnog isključivanja više dretvi uz pomoć nedjeljive instrukcije TAS/SWAP/FATCH_AND_ADD. Koja je prednost tih rješenja u odnosu na Lamportov algoritam međusobnog isključivanja?

```
dok je (1){
      TAS ZASTAVICA;
      dok je (ZASTAVICA != 0){
            TAS ZASTAVICA;
      }
      kritični odsječak;
      ZASTAVICA=0;
      nekritični odsječak;
}
```

Rješenja sa nedjeljivim instrukcijama su jednostavnija, kraca i zahtjevaju manje varijabli, pa se zbog toga brze izvode i zauzimaju manje memorije.

42. Koji je najveći zajednički nedostatak algoritmima međusobnog isključivanja (Dekkerov, Petersonov, Lamportov te algoritmima ostvarenim uz pomoć sklopovske potpore).

To što dretve izvode radno čekanje i time beskorisno troše vrijeme svojeg procesora i sabirničke cikluse.

5. Jezgra operacijskog sustava

43. Što predstavlja pojam „ulazak u jezgru“ i kada se zbiva?

Pojam predstavlja poziv jezgrine funkcije i zbiva se kada se dogodi prekid.

44. Na što se svodi „izlazak iz jezgre“?

Izlazak se svodi na pokretanje jedne od dretvi, pri čemu procesor mora biti vraćen u korisnički način rada.

45. Navesti izvore prekida u jednostavnom modelu jezgre.

Sklopovski prekidi, programski prekidi i prekidi od sata.

46. Od čega se sastoji jezgra operacijskog sustava?

Od struktura podataka i jezgrinih funkcija.

47. Navesti sadržaj opisnika dretve.

Kazaljka, PID, ID dretve, stanje dretve, prioritet, početna adresa dretvenog spremničog prostora, adresa prve instrukcije, kašnjenje, prostor za smještaj konteksta, veličina dretvenog adresnog prostora (10).

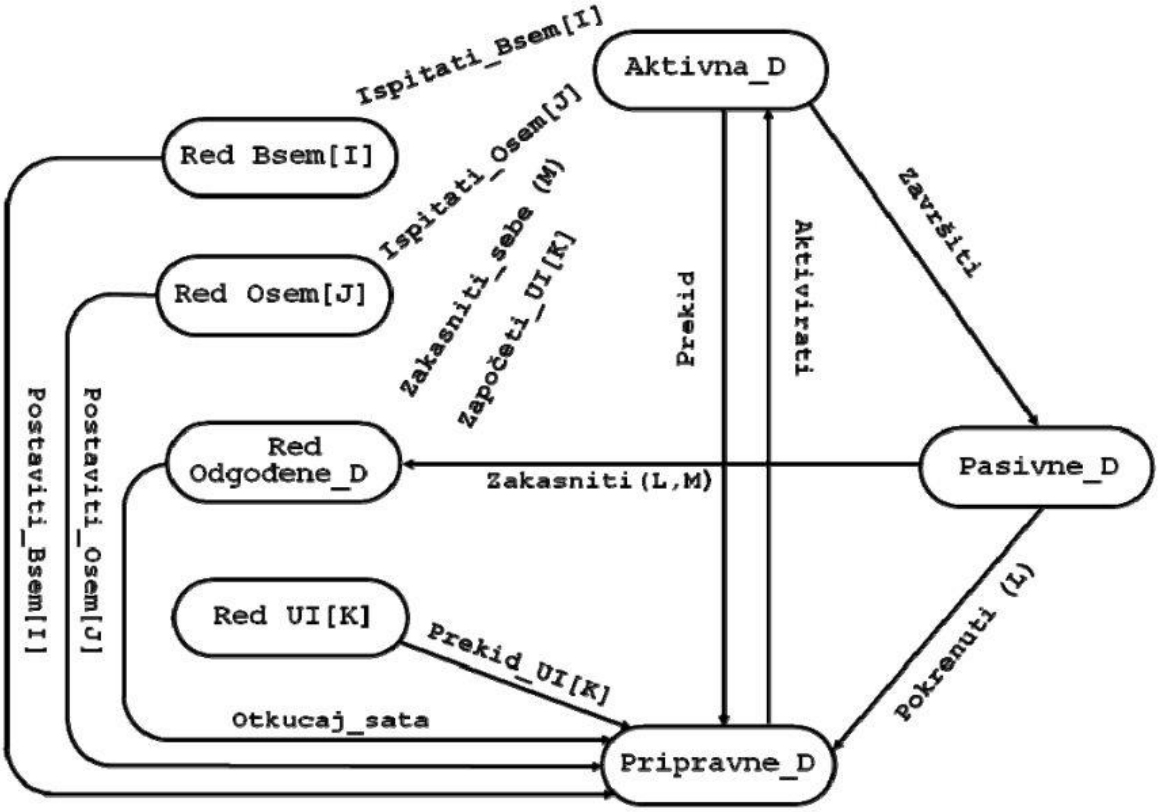
48. Navesti strukture podataka jezgre.

Liste i opisnici dretvi.

49. Koja su blokirana stanja dretvi?

4. stanja: čekanje na binarnom i općem semaforu, čekanje na istek kašnjenja, čekanje na UI operaciju

50. Skicirati graf mogućih stanja dretvi.



51. Što obavlja instrukcija „aktivirati prvu dretvu iz reda Pripravne_D“?

- Premješta prvi opisnik iz reda PRIPRAVNE D u red AKTIVNA D
- Obnovlja kontekst opisnika AKTIVNA D
- Omogućava prekidanje

52. Što obavlja instrukcija „vratiti se iz prekidnog načina“?

Vraća u procesor sadržaj PC i prebacuje procesor iz sustavskog u korisnički način rada.

53. Čemu služe jezgrini mehanizmi binarni i opći semafor?

Služe za sinkronizaciju dretvi.

54. Koje strukture podataka koriste BSEM, OS i OSEM?

BSEM koristi varijablu Bsem[I].v i kazaljku. OS koristi varijablu OS.v i kazaljku. OSEM koristi varijablu Osem[J].v i kazaljku.

55. U pseudokodu napisati jezgrine funkcije Čekaj_BSEM, Postavi_BSEM, Čekaj_OS, Postavi_OS, Čekaj_OSEM i Postavi_OSEM.

BSEM

```
Čekaj_Bsem(i) {
      pohraniti kontekst u opisnik Aktivna_D;
      premjestiti opisnik iz Aktivna_D;
      ako je ((Bsem[i].v)==0) && (red Bsem[i] neprazan)){
            premjestiti prvi opisnik iz reda Bsem[i] u redu
            Pripravne_D;
      }
      Inače {
            Bsem[i].v==1;
      }
      Aktivirati prvu dretvu iz reda Pripravne_D;
}

Postaviti_Bsem (i) {
      pohraniti kontekst u opisnik Aktivna_D;
      premjestiti opisnik iz reda Aktivna_D u red Pripravne_D;
      ako je ((Bsem[i].v==0) && (red Bsem[i] neprazan)) {
            premjestiti prvi opisnik iz reda Bsem[i] u red
            Pripravne_D;
      }
      Inače {
            Bsem[i].v = 1;
      }
      aktivirati prvu dretvu iz reda Pripravne_D;}
```

OS

```
Čekaj_OS (j) {
      Pohraniti kontekst u opisnik Aktivna_D;
      OS[j].v = Os[j].v – 1;
      ako je (Os[j].v >= 0) {
            obnoviti kontekst iz opisnika Aktivna_D;
            omogućiti prekidanje;
            vratiti se iz prekidnog načina;
      }
      Inače {
            premjestiti opisnik iz reda Aktivna_D u red Os[j];
            aktivirati prvu dretvu iz reda Pripravne_D;
      }
}
```

OSEM

```
Čekaj_Osem[j] {
      pohraniti kontekst u opisnik Aktivna_D;
      ako je (Osem[j].v >= 1) {
            Osem[j].v = Osem[j].v – 1;
            obnoviti kontekst iz opisnika Aktivna_D;
            omogućiti prekidanje;
            vratiti se iz prekidnog načina;
      }
      inače {
            premjestiti opisnik iz reda Aktivna_D u red Osem[j];
            aktivirati prvu dretvu iz reda Pripravne D;
      }
}
```

```
Postaviti_Osem[j] {
      pohraniti kontekst u opisnik Aktivna_D;
      premjestiti opisnik iz reda Aktivna_D u redu Pripravne_D;
      ako je ((Osem[j].v == 0) && (red Osem[j] neprazan)) {
            premjestiti prvi opisnik iz reda Osem[j] u red
            Pripravne_D;
      }
      inače {
            Osem[j].v == Osem[j].v + 1;
      }
      aktivirati prvu dretvu iz reda Pripravne_D;
}
```

56. Opisati način umetanja opisnika dretve u listu Zakašnjele_D. Koja vrijednost se upisuje u polje Zadano_kašnjenje u opisniku dretve?

57. Koje vrste prekida uzrokuju jezgrine funkcije Započeti_UI i Prekid_UI u jednostavnom modelu jezgre?

Sklopovski prekid.

58. Može li se prekinuti dretva koja obavlja neku jezgrinu funkciju?

Ne, jezgrina funkcija ne može biti prekinuta.

59. Na koji način se jezgrine funkcije obavljaju međusobno isključivo na jednoprocesorskom računalu, a kako na višeprocesorskom računalu?

(Jednoprocesorska računala) Na način da se jezgrine funkcije pozivaju sklopovskim ili programskim prekidom.

Na višeprocesorskom računalu struktura podataka jezgre se mora nalaziti u dijeljenom spremniku. Ona mora biti dohvatljiva svim dretvama bez obzira u kojim se procesorima dretve izvodile. Do te strukture jezgrine funkcije moraju pristupati međusobno isključivo.