OPERACIJSKI SUSTAVI

PITANJA ZA VJEŽBU – 1. CIKLUS

(by kate)

1. Uvod

1. Što je Operacijski sustav?

Skup programa koji omogućavaju izvođenje osnovnih operacija na računalu – potpora raznovrsnim primjenskim rogramima.

2. Navesti osnovne dijelove operacijskog sustava.

- Upravljanje datotečnim sustavom
- upravljanje spremnikom (memorijom)
- upravljanje U/I uređajima
- API (Application programming interface)
- GUI (Graphical user interface)
- procesi i dretve (komunikacija, sinkronizacija i raspoređivač poslova), mrežni i sigurnosni podsustav.

3. Program čiji je izvorni kod u datoteci lab5.c, kompajlira se sa: _____gcc lab5.c_____, a pokrece sa __./a.out_____ .

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

Pogreska kod inicijalizacije polja, pointera ili varijabli (neinicijailzacija ili adresirano izvan adresnog prostora. (NEZNAM!)

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

Sučelje je utvrđeni način komunikacije (čvrsto dogovoreni način uspostavljanja veze između dvije inače nerazdvojne cjeline; npr. Komunikacija između korisnika i operacijskog sustava).
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

2. Model jednostavnog računala

6. Čime su određena svojstva i ponašanje procesora?

Skupom registara (služe za pohranjivanje svih sadržaja koji ulaze i izlaze iz procesora i u njemu se transformiraju) i skupom instrukcija (određen izvedbom ALU i upravljačke jedinke procesora).

7. Navesti osnovni skup registara procesora.

Adresni međuregistar
Podatkovni međuregistar

Instrukcijski registar
Programsko brojilo (PC)
Registar kazaljke stoga (SP)
Registar stanja (SR)
Registri opće namjene (R)

8. Što je to sabirnički ciklus?

Period u kojem se obavlja jedno čitanje ili pisanje u radni spremnik.

9. U pseudokodu napisati što procesor trajno radi?

ponavljati {
dohvatiti iz spremnika instrukciju na koju pokazuje programsko brojilo;
dekodirati instrukciju, odrediti operaciju koju treba izvesti;
povećaj sadržaj programskog brojila tako da pokazuje na sljedeću instrukciju;
odrediti odakle dolaze operandi i kuda se pohranjuje rezultat;
operande dovesti na aritmetičko-logičku jedinicu, izvesti zadanu operaciju;
pohraniti rezultat u odredište;
} dok je (procesor uključen);

10. Što je kontekst dretve?

Sadržaj trenutne dretve pohranjen u registrima procesora; Svi registri osim programskog brojila.

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 izvest;
Povečati programsko brojilo, da pokazuje na sljedeću instrukciju;
Ako je (dekodirana instrukcija poziv potprograma) {
Pohraniti sadržaj programskog brojila na stog;
Smanjiti sadržaj registra SP, tako da pokazuje na sljedeće prazno mjesto;
Iz adresnog dijela instrukcije odrediti adresu početka potprograma;
Staviti adresu u programsko brojilo;
}
Inače
Obaviti instrukciju na način određen dekodiranim operacijskim kodom;
}
Dok je (procesor uključen);

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

Program – statični niz instrukcija, pohranjen na papiru, disketi, memoriji itd.
Dretva – niz instrukcija koje se izvode i kontroliraju proces.
Proces – program u izvođenju; skup računalnih resursa koji omogućuju izvođenje programa; okolina u kojoj se program izvodi; sve što je potrebno za izvođenje programa; sastoji se od:

- barem jedne dretve
- zajedničkog adresnog prostora
- adresnog prostora rezerviranog za svaku pojedinu dretvu

- stog, kazaljke stoga, opisnici datoteka, opisnici cjevovoda, redovi poruka, semafori, uvjetne varijable, Zaključavanja.

13. Kako je moguć višeprogramski rad na jednoprocesorskom računalu?

Tako da se svaka dretva izvodi naizmjence pa dobijemo privid istovremenosti. Ključna je pravilna izmjena konteksta dretve.

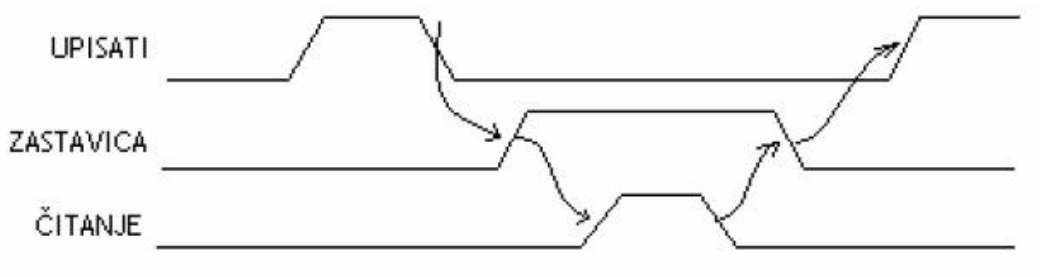
3. Obavljanje UI operacija, prekidni rad

14. Skicirati način spajanja UI naprave na sabirnicu. (03-04)

15. Što je radno čekanje?

Režim rada procesora u kojem procesor čeka na određeni događaj u programu (npr. pojava zastavice) i troši vrijeme dok se on ne dogodi.

16. Skicirati signale dvožičnog rukovanja.



Podatak se upisuje kada je zastavica u niskoj razini, nakon upisivanja se ona podiže i tada se aktivira čitanje PR koje po svom završetku spušta zastavicu i omogučuje novo čitanje.
S obzirom da se signali na dva vodiča ("žice") naizmjence podižu i spuštaju taj je protokol nazvan dvožičnim rukovanjem (engl. Twowire handshaking).

17. Što se zbiva kada se dogodi prekid?

Pojava prekidnog signala prebacuje procesor u tzv. sustavski (jezgreni) način rada

1. zabranjuje prekidanje

2. prebacuje adresiranje u sustavski adresni prostor
3. aktivira sustavsku kazaljku stoga
4. pohranjuje programsko brojilo na sustavski stog
5. u programsko brojilo stavlja adresu potprograma za obradu prekida

U prekidnom potprogramu:

1. pohraniti kontekst
2. posluživanje prekida
3. obnavlja kontekst
4. omogući prekidanje
5. vrati se iz prekidnog načina rada (vračanje programskog brojila sa stoga, prebacivanje adresiranja u korisnički adresni prostor, aktiviranje korisničkog registra kazaljke stoga)

18. Kako treba nadopuniti ponašanje procesora da on omogučuje 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 čekanja $K\_Z[i] = 1$*;*

Poništi zastavicu u registru stanja prekida i;

Dok je $((postoji\ K\_Z[j]) != 0) \wedge (J > T\_P))${

Odabrati najveći j;

$K\_Z[j] = 0$*;*

Pohraniti kontekst sa sustavskog stoga i T_P *u* $KON[j]$*;*

$T\_P = j$

Omogući prekidanje;

Pređi u korisnički način rada;

Pozovi potprogram za obradu prekida j;

Zabrani prekidanje;

Pređi u sustavski način rada;

Vrati na sustavski stog i u varijablu T_P sadržaj iz KON[j];

}

}

Obnovi kontekst sa sustavskog stoga;

Omogući prekidanje;

Vrati se iz prekidnog načina rada;

}

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

Pohraniti kontekst – stavlja na sustavski stog sadržaje svih registara osim programskog brojila

Vratiti se iz prekidnog nacina – vraća programsko brojilo sa stoga; prebacuje adresiranje u korisnički adresni prostor, aktivira korisnicki registar kazaljke stoga

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

Pohraniti kontekst

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

Procesor se prebacuje na rutinu za obradu prekida tako da promijeni PC na adresu te rutine. Što znači da ta rutina ne može pohraniti PC (jer je već "uništen") već to mora učiniti procesor. Također, vraćanje PCa pokreće prekinutu dretvu!

22. Što se zbiva kada obrada nekog prekida završi?

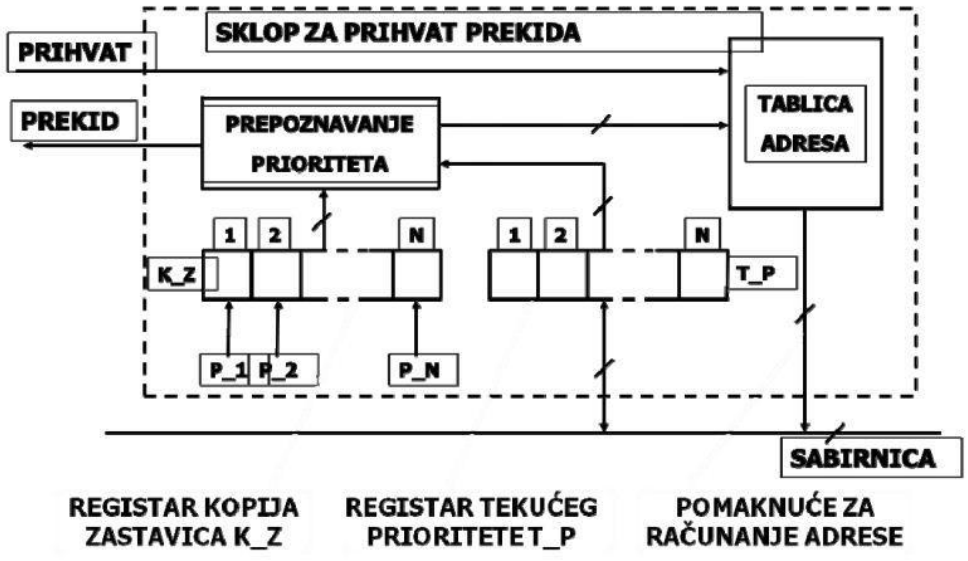
- Obnavlja se kontekst sa s sustavskog stoga;
- omogučuje se prekidanje (odgodjeno do upisa nove vrijednosti u PC);
- prebacuje se adresiranje u korisnički prostor;
- vrača se iz prekidnog načina rada

23. Koje strukture podataka treba sadržavati operacijski sustav koji omogućuje prihvat prekida razlicitih prioriteta?

- Varijablu T_P (tekući prioritet) – broj tekućeg prioriteta dretve koja se upravo izvodi
- Polje KON[n] (polje za kontekst) – pohranjuje se kontekst dretve i T_P
- Polje K_Z[n] (kontrolna zastavica) – polje u kojem su zapisani prekidi koji čekaju na obradu (1-stigao prekid, 0-nije)

24. Opisati sklop za prihvat prekida.

Sastoji se od sklopa za prepoznavanje prioriteta, tablice adresa i dva registra: K_Z i T_P. Iz njega izlazi signal PREKID, a ulazi signal PRIHVAT U K_Z se upisuje jedinica kada pripadni pristupni sklop zatraži prekid. U T_P se zapisuje prioritet dretve koju procesor upravo izvodi, jedinicom u odgovarajućem bitu registra. Sadržaj ta dva registra dovode se na sklop za prepoznavanje prioriteta. Taj sklop propusta prekid prema procesoru samo onda kada je prioritet zahtjeva veći od onog zabilježenog u registru T_P. Istodobno s propuštanjem prekidnog signala prema procesoru mora se obrisati bit u registru K_Z. Tablica adresa sadrži adrese (ili pomaknuća za određene adrese) na kojima počinje prekidni program koji je propušten prema procesoru. Sadržaj tablice se pod utjecajem signala PRIHVAT prenosi na sabirnicu odakle ga procesor može dohvatiti i neposredno oblikovati adresu na koju treba skočiti.



25. Kako treba nodopuniti ponašanje procesora da on omogucuje 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.

Pokusaj djeljenja s nulom, adresiranje nepostojece lokacije u adresnom prostoru, dekodiranje nepostojece instrukcije i sl.

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

Događaju se kad se dogodi prekid. Pozivaju se neke vrste potprograma koje se izvode u jezgrenom (sustavskom) nacinu rada (jezgrine funkcije).

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

Adresni registar (AR) - sprema se početna adresa bloka koji se želi prenesti

Brojač (BR) - sprema se broj znakova koji se prenose

Registar stanja (RS) - daje procesoru zahtjev za prekid

podatkovni registar (PR) - spremaju se podaci koji se prenose

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

Dok je $(BR > 0)${

Zatraziti sabirnicu;

Cekati na dodjelu sabirnice;

Postaviti na adresni dio sabirnice sadrzaj registra AR;

Prenjeti na tu adresu sadržaj podatkovnog registra PR (ili obrnuto);

$AR++;$

$BR--;$

}

Postaviti signal PREKID;

30. Opisati cvrsto povezani višeprocesorski sustav.

(slika 03-83)

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. Medusobno iskljucivanje u višedretvenim sustavima

31. Što je zajednicko procesu roditelju i procesu djetetu? Koje racunalne resurse dijele dretve istog procesa?

Roditejlu i djetetu nista nije zajednicko. Proces dijete je kopija procesa roditelj, s toga ono ima iste instrukcije i podatke, ali svaki proces ima svoj adresni prostor i ne mogu jedno drugom adresirat varijable.
Dretve istog procesa dijele sve računalne resurse.

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

Procesni prostor: više dretvenih prostora i zajednički prostor (koji mogu dohvaćati sve dretve procesa – globalne varijable).
Dretveni prostor: dio za instrukcije dretve, dio za stog dretve i dio za lokalne podatke dretve

33. Navesti uvjet nezavisnosti dretvi.

$$(X_i \cap Y_j) \cup (X_j \cap Y_i) \cup (Y_i \cap Y_j) = 0$$

34. Navesti uvjete koje mora zadovoljavati algoritam medusobnog iskljucivanja dretvi.

- Dretve se odvijaju međusobno isključivo (dvije dretve ne smiju obavljati K.O.)
- Algoritam mora funkcionirati i onda kada su brzine izvođenja dretvi različite
- Ako neka dretva zastane u N.K.O. to ne smije spriječiti drugu dretvu da uđe u K.O.
- Izbor koja dretva će ući u K.O. mora se zbiti u konačnom vremenu.

35. Za zadani algoritam medusobnog iskljucivanja ustanoviti je li ispravan. Obrazložiti odgovor.

```
Dretva I{
      dok je (1){
            dok je (ZASTAVICA[J] != 0);
            ZASTAVICA[I] = 1;
            kriticni odsjecak;
            ZASTAVICA[I] = 0;
            nekriticni odsjecak;
      }
}
```

Nije ispravan. Dretva je u K.O kada joj je zastavica postavljena u 1. Kada dretva Di krece u svoj K.O. dretva Dj ce isto biti u K.O. jer ce Di uci u petlju tek kada je zastavica od Dj=1 sto znaci da Dj u K.O. S druge strane, ako Dj nije u K.O.Di takodjer nece uci u njega.

36. Cemu služi Dekkerov, a cemu Lamportov algoritam? Koje strukture podataka koriste?

Dekkerov algoritam – algoritam medjusobnog iskljucivanja za dvije dretve

Strukture podataka: varijable ZASTAVICA[I], ZASTAVICA [J] i PRAVO

Lamportov algoritem – algoritam medjusobnog iskljucivanja za n dretvi

Strukture podataka: polje zastavica (koje govori koja dretva pokušava ući) ULAZ[I], polje brojeva dretvi BROJ[J] ivarijabla zadnjeg broja ZADNJI_BROJ.

37. Navesti Dekkerov/Lamportov algoritam.

Dekkerov algoritam

dok je (1) {

$ZASTAVICA[i] = 1;$

Čitati varijablu $ZASTAVICA[j];$

Dok je ($ZASTAVICA[j]! = 0$){

Ako je ($PRAVO\ != \ i$){

$ZASTAVICA[i] = 0;$

Dok je (PRAVO !=i){

Čitati varijablu PRAVO;

}

$ZASTAVICA[i] = 1;$

}

Čitati varijablu $ZASTAVICA[j]$

Kritični odsječak;

PRAVO=j;

$ZASTAVICA[i] = 0;$

Nekriticni odsječak;

}

Lamportov algoritam

Dok je (1){

$ULAZ[i] = 1;$

Čitati ZADNJI_BROJ;

$BROJ[i] = ZADNJI\_BROJ + 1;$

$ZADNJI\_BROJ = BROJ[i];$

$ULAZ[i] = 0;$

Za ($j = 0;\ j < n; j++$){

Čitati varijablu ULAZ[j];

Dok je ($ULAZ[j] == 1$){

Čitati varijablu ULAZ[j];

}

Čitati varijablu BROJ[j];

Dok je ($BROJ[j]! = 0$) && $((BROJ[j], j) < (BROJ[i], i)$){

Čitati varijablu BROJ[j];

}

}

Kritični odsječak;

$BROJ[i] = 0;$
Nekritični odsječak;
}

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 nacin medusobnog iskljucivanja više dretvi na jednoprocesorskom racunalu?

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 medusobnom iskljucivanju.

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 medusobnog iskljucivanja više dretvi uz pomoc nedjeljive instrukcije TAS/SWAP/FATCH_AND_ADD. Koja je prednost tih rješenja u odnosu na Lamportov algoritam medusobnog iskljucivanja?

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

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

42. Koji je najveci zajednicki nedostatak algoritmima medusobnog iskljucivanja (Dekkerov, Petersonov, Lamportov te algoritmima ostvarenim uz pomoc sklopovske potpore).

Dretve koje žele ući u K.O. izvode radno čekanje. Time beskorisno troše vrijeme svojih 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 prekidom.

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

Na aktiviranje jedne od dretvi, pri čemu procesor mora biti vračen u korisnički način rada.

45. Navesti izvore prekida u jednostavnom modelu jezgre.

- ulazno-izlazne naprave (sklopovski prekid)
- sat
- dretve (programski prekid)

46. Od cega se sastoji jezgra operacijskog sustava?

- Strukture podataka jezgre
- Jezgrenih funkcija

47. Navesti sadržaj opisnika dretve.

- Kazaljka ili više njih za premještanje iz liste (reda) u listu(red)
- Identifikacijski broj procesa kojoj dretva pripada (PID)
- Identifikacijski broj dretve (ID)
- Stanje dretve (pasivna, aktivna, blokirana, pripravna)
- Prioritet (mjesto gdje je zapisan prioritet)
- početna adresa dretvenog adresnog prostora
- veličina dretvenog adresnog prostora
- adresa prve instrukcije dretve
- zadano kašnjenje
- prostor za smještanje konteksta (u kojem se nalazi programsko brojilo)

48. Navesti strukture podataka jezgre.

- Liste:

1. **Pasivne_D** – kada se dretva nalazi na samo jednoj listi (postojece_D), onda je u pasivnom stanju
2. **Aktivna_D** – dretve koje se izvode; broj članova u toj listi jednak je broju procesora.
3. **Pripravne_D** – ako se ne izvode, a spremne su. Prema načinu formiranja red može biti – po redu prispjeća ili prioritetni
4. **Red BSEM[i]** – dretve koje cekaju na binarnom semaforu
5. **Red OSEM[j]** – dretve koje cekaju na opčem semaforu
6. R**ed odgođen1_D –** zadano kašnjenje dretve
7. **Red UI[k]** – ima ih koliko ima U/I naprava

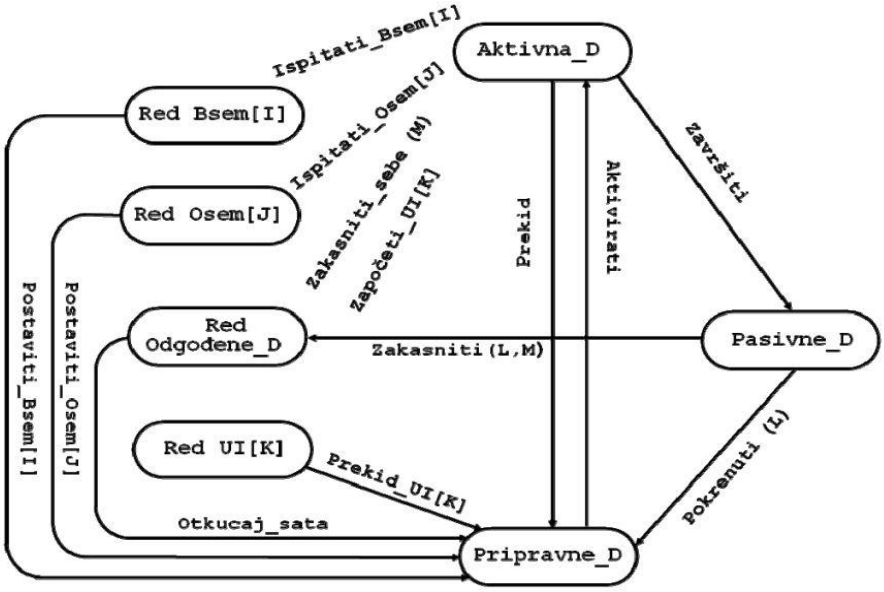
- Opisnici dretvi

49. Koja su blokirana stanja dretvi?

- Čekanje na binarnom seamforu
- Čekanje na općem semaforu

- Čekanje na istek zadanog intervala kašnjenja (odgođene dretve)
- Čekanje na završetak U/I operacije.

50. Skicirati graf mogucih stanja dretvi.(05-35)



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

Premjestiti prvi opisnik iz reda Pripravne_D u red Aktivne_D;
Obnoviti kontekst iz opisnika Aktivna_D;
Omogučiti prekidanje;
Vratiti se iz prekidnog načina;

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

Vraća u procesor sadržaj programskog brojila i prevodi procesor iz sustavskog u korisnički način rada.

53. Cemu služe jegrini mehanizmi binarni i opci semafor?

Za međusobno iskljucivanje, odnosno sinkronizaciju dretvi.

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

Varijablu sa stanjem semafora i kazaljku na listu dretvi koje cekaju na semafor.
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 Cekaj_BSEM, Postavi_BSEM, Cekaj_OS, Postavi_OS, Cekaj_OSEM i Postavi_OSEM.

Cekaj_BSEM

```
j-funkcija Cekaj_BSEM [I]{
        pohrani kontekst u opisnik Aktivna_D;
        Ako je (BSEM[I].v==1){
                BSEM[I].v ==0;
                Obnovi kontekst iz opisnika Aktivna_D;
```

```
            Omoguci prekidanje;
            Vrati se iz prekidnog načina rada;
        }
        Inače{
            Premjesti opisnik iz Aktivna_D u red BSEM[I];
            Aktiviraj prvu dretvu iz reda Pripravne_D;
        }
}
```

Postavi_BSEM

```
j-funkcija Postavi_ BSEM[I]{
        pohrani kontekst u opisnik Aktivna_D;
        premjesti opisnik iz reda Aktivna_D u red Pripravna_D;
        Ako je ((BSEM[I].v ==0)&&(red BSEM[I] nije prazan)){
            Premjesti prvi opisnik iz reda BSEM[I] u red Pripravne_D;
        }
        Inače{
            BSEM[I].v=0;
        }
        Aktiviraj prvu dretvu iz reda Pripravne_D;
}
```

Cekaj_OS

```
j-funkcija Cekaj_OS [J]{
        pohrani kontekst u opisnik Aktivna_D;
        OS[J].v -- ;
        Ako je (OS[J].v >=0){
            Obnovi kontekst iz opisnika Aktivna_D;
            Omoguci prekidanje;
            Vrati se iz prekidnog načina rada;
        }
        Inače{
            Premjesti opisnik iz Aktivna_D u red OS[J];
            Aktiviraj prvu dretvu iz reda Pripravne_D;
        }
}
```

Postavi_OS

```
j-funkcija Postavi_OS[J]{
        pohrani kontekst u opisnik Aktivna_D;
        premjesti opisnik iz reda Aktivna_D u red Pripravna_D;
        OS[J].v ++;
        Ako je ((OS[J].v==0)&&(red OS[J] nije prazan)){
            Premjesti prvi opisnik iz reda OS[J] u red Pripravne_D;
```

```
        }
        Aktiviraj prvu dretvu iz reda Pripravne_D;
}
```

Cekaj_OSEM

```
j-funkcija Cekaj_OSEM [J]{
        pohrani kontekst u opisnik Aktivna_D;
        Ako je (OSEM[J].v >=1){
                OSEM[J].v --;
                Obnovi kontekst iz opisnika Aktivna_D;
                Omoguci prekidanje;
                Vrati se iz prekidnog načina rada;
        }
        Inače{
                Premjesti opisnik iz Aktivna_D u red OSEM[J];
                Aktiviraj prvu dretvu iz reda Pripravne_D;
        }
}
```

Postavi_OS

```
j-funkcija Postavi_OS[J]{
        pohrani kontekst u opisnik Aktivna_D;
        premjesti opisnik iz reda Aktivna_D u red Pripravna_D;
        Ako je ((OSEM[J].v==0)&&(red OSEM[J] nije prazan)){
                Premjesti prvi opisnik iz reda OSEM[J] u red Pripravne_D;
        }
        Inače{
                OS[J].v ++;
        }
        Aktiviraj prvu dretvu iz reda Pripravne_D;
}
```

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

Lista Zakašnjele_D je složena prema vremenima kašnjenja. Prva dretva u listi ima najmanje vrijeme spavanja, dok zadnja ima najdulje.

U polje Zadano_kašnjenje prvog opisnika upisuje se apsolutna virjednost zadanog kašnjenja, a u ostale opisnike samo dodatno odgađanje u odnosu na prethodnu dretvu.

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

Sklopovske prekide.

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

Ne, jezgrina funkcija ne može biti prekinuta.

59. Na koji nacin se jezgrine funkcije obavljaju medusobno iskljucivo na jednoprocesorskom racunalu, a kako na višeprocesorskom racunalu?

Međusobno isključivanje na jednoprocesorskom računalu ostvaruje se prekidima. Na višeprocesorskom računalu struktura podataka jezgre se mora nalaziti u dijeljenom spremniku. Za sinkronizaciju pristupa strukturama podataka jezgre koristi se zastavica OGRADA_JEZGRE, a međusobno isključivanje se ostvaruje radnim čekanjem.