Operacijski sustavi

[Rješenja pitanja za ponavljanje]

By Benko & pcurko10

1.1 Što je operacijski sustav?

Operacijski sustav je skup osnovnih programa koji omogućuju provođenje radnih zadataka na računalu – izvođenje operacija računala. Osnovni programi služe kao potpora različitim primjenskim programima (preko kojih korisnici obavljaju korisne zadatke) koji transformiraju računalni sustav u virtualni stroj.

1.2 Koji su osnovni zadaci operacijskog sustava?

- olakšavanje uporabe računala
- djelotvorno iskorištavanje svih djelova sustava
- omogućiti višeprogramski rad
- dodjeljivanje sredstava programima
- omogućiti komunikacije između više računala

1.3 Navesti osnovne djelove operacijskog sustava.

- procesi
- dretve
- raspoređivač poslova
- komunikacija
- sinkronizacija
- API
- GUI
- sigurnosni podsustav
- mrežni podsustav
- upravljanje I/O uređajima
- upravljanje memorijom
- upravljanje datotečnim sustavom

1.4 Što je to sučelje?

Sučelje je čvrsto dogovoreni način uspostavljanja veze između razdvojenih cjelina – način postavljanja zahtjeva operacijskom sustavu i izgled povratnih poruka operacijskog sustava.

1.5 Što je to sučelje primjenskih programa (API)?

API je skup operacija koji omogućuje programerima primjenskih programa djelotvornije korištenje računalnog sustava. Operacijski sustav kroz API oslobađa programere detalje oko pristupa računalnim sredstvima.

2.1 Čime su određena svojstva i ponašanje procesora?

Svojstva procesora su određena skupom registara i skupom instrukcija na temelju kojih upravljačka jedinica određuje koju će operaciju izvesti aritmetičko-logička jedinica.

2.2 Navesti osnovni skup registara procesora.

ADRESNI REGISTAR	postavlja se adresa na adresnu sabirnicu
PODATKOVNI REGISTAR	podaci se prenose iz CPUa u spremnik i obrnuto
INSTRUKCIJSKI REGISTAR	prenosi se instrukcija dobivena iz spremnika
PROGRAMSKO BROJILO	sadržava adresu sljedeće instrukcije
REGISTAR KAZALJKE STOGA	poseban način adresiranja spremnika
STATUS REGISTAR	zapisivanje zastavica
REGISTRI OPĆE NAMJENE	pohranjivanje operanada i rezultata

2.3 Što je to sabirnički ciklus?

Vrijeme se na sabirnici djeli na sabirničke cikluse. U jednom sabirničkom ciklusu ostvaruje se jedna veza – prijenos sadržaja

2.4 U pseudokodu napisati što procesor trajno radi.

```
Ponavljati {
    Dohvatiti iz spremnika instrukciju na koju
        pokazuje PC;
    Dekodirati instrukciju, odrediti koju operaciju
        izvesti;
    Povećati sadržaj PCa;
    Odrediti odakle dolaze operandi i gdje se
        pohranjuje rezultat;
    Dovesti operande na ALU i izvesti zadanu
        operaciju;
    Pohraniti rezultat na odredište;
} Dok je (Procesor uključen);
```

2.5 Što je kontekst dretve?

Sadržaj registara procesora zove se kontekstom dretve.

2.6 Što se zbiva pri izvođenju instrukcije za poziv potprograma?

1) pohranjuje se sadržaj PCa na STOG
2) smanjuje se sadržaj SPa tako da pokazuje na sljedeće prazno mjesto
3) odrediti početnu adresu potprograma i smjestiti je u PC

2.7 Definirati osnovne pojmove: program, proces, dretva.

PROGRAM statički niz instrukcija u memoriji

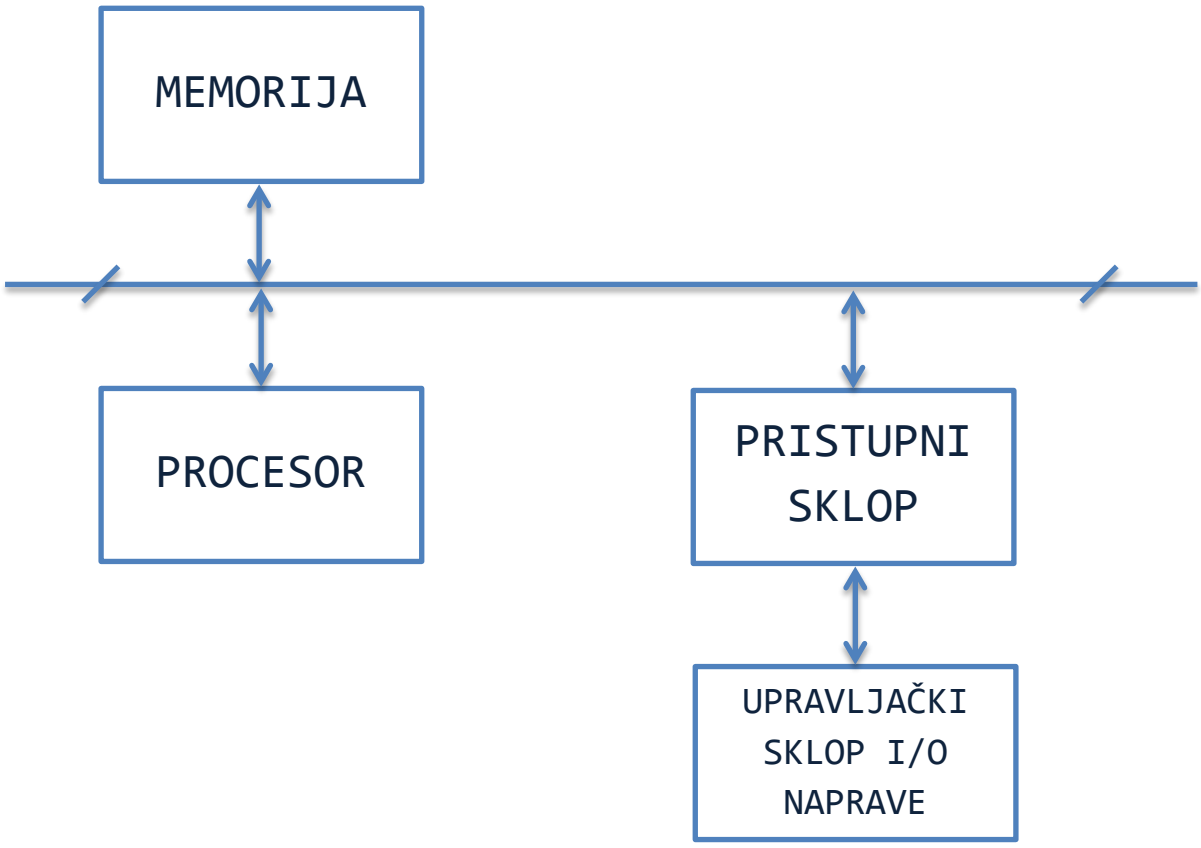
PROCES program u izvođenju (sve što je potrebno da bi se neki program mogao izvoditi)

DRETVA niz instrukcija koji se izvodi u vremenu (dinamički)

2.8 Kako je moguć višeprogramski rad na jednoprocesorskom računalu?

Višeprogramski rad moguć je tako da jedini procesor "provlači" jednu od više dretvi, time se postiže da procesor izvodi jednu od dretvi za vrijeme dok druge moraju zbog nekog razloga čekati.

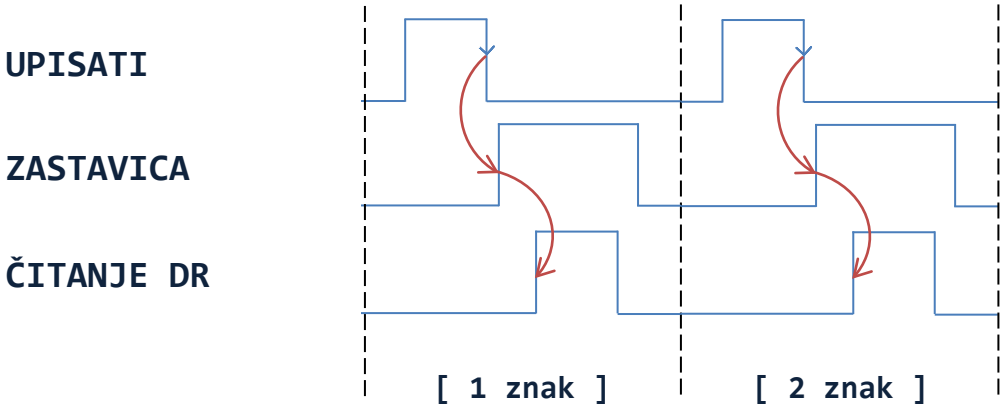
3.1 Skicirati način spajanja UI naprave na sabirnicu.



3.2 Što je radno čekanje?

Radno čekanje je kada procesor uzastopce čita registar stanja i ispituje je li program/sklop spreman.

3.3 Skicirati signale dvožičnog rukovanja.



3.4 Što se zbiva kada se dogodi prekid?

Pojava prekidnog signala prebacuje procesor u tzv. ″sustavski način rada″ (jezgreni način rada – *kernel mode*).

3.5 Kako treba nadopuniti ponašanje procesora da on omogućuje prekidni rad bez sklopa za prhivat prekida?

Model pristupnog sklopa mora biti moificiran tako da podizanje bita ZASTAVICA u registru stanja popraćeno generiranjem električnog signala koji se preko posebne žice dovodi do procesora (taj signal govori procesoru da je došlo do prekida).

Ponašanje procesora pri izvođenju svake instrukcije treba nadopuniti tako da on na kraju svake instrukcije ispituje jeli došlo do prekida.

3.6 Pojasniti instrukcije ″POHRANITI_KONTEKST″ i ″VRATITI_SE_IZ_PREKIDNOG_NAČINA″.

POHRANITI_KONTEKST pohranjivanje sadržaja svih registara na sustavski stog

VRATITI_SE_IZ_PREKIDNOG_NAČINA instrukcija koja prebaciva adresiranje u korisnički adresni prostor i aktivira korisnički SP

3.7 Što treba načiniti na početku svakog potprograma za obradu prekida?

Potrebno je privremeno onemogućiti daljnje prekidanje.

3.8 Zašto se programsko brojilo tretira zasebno prilikom pohrane konteksta?

Pohrana konteksta se odvija u prekidnom potprogramu – zbog toga je potrebno prije odlaska u potprogram pohraniti PC jer bi se inače (pravi) PC izgubio pošto bi u njega pohranili adresu prekidnog potprograma.

3.9 Što se zbiva kada obrada nekog prekida završi?

Nakon završetka obrade prekida, ponovno se pokreće prekinuta dretva korisničkog programa.

3.10 Koje strukture podataka treba sadržavati operacijski sustav koji omogućuje prihvat prekida različitih prioriteta?

- varijabla T_P (tekući prioritet)
- polje KON[N] (polje u koje će se pohranjivati kontekst pojedine dretve)
- polje zastavica K_Z [N] (kopije zastavica)

3.11 Opisati sklop za prihvat prekida.

Sklop bi trebao prema procesoru propustiti samo onaj prekidni signal koji ima veći prioritet od dretve koju procesor upravo izvodi. Sklop sadrži dva registra

K_Z (kopije zastavica)
T_P (tekući prioritet)

3.12 Kako treba nadopuniti ponašanje procesora da on omogućuje prekidni rad sa sklopom za prihvat prekida?

Procesor bi u trenutku kada prihvaća prekid trebao postaviti signal PRIHVAT koji će djelovati na sklop za prihvat prekida.

3.13 Navesti koje sve radnje mogu generirati prekide unutar procesora.

- nepostojeća lokacija u adresnom prostoru
- nepostojeća instrukcija
- djeljenje s 0
- ....i slično

3.14 U kojem će se slučaju dogoditi *"poziv jezgre"*, odnosno *"ulazak u jezgru"* i što se tada poziva?

Dogoditi će se u slučaju pojave prekida. Tada se pozivaju određene jezgrine funkcije.

3.15 Navesti osnovne registre pristupnog sklopa za neposredni pristup spremniku (DMA).

DMA sadrži sljedeće registre:

- *data register* (DR)
- *status register* (SR)
- *address register* (AR)
- *counter* (CNT)

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

```
Dok je (BR > 0) {
    Zatražiti sabirnicu;
    Čekati na dodjelu sabirnice;
    Postaviti na adresnu sabirnicu registar AR;
    Prenijeti na tu adresu sadržaj DR (ili obrnuto);
    AR = AR + 1;
    BR = BR – 1;
}
```

3.17 Opisati čvrsto povezani višeprocesorski sustav.

Višeprocesorski sustav koristi više procesora ali zajednički spremnik i zajedničku sabirnicu. Struktura podataka nalazi se u zajedničkom spremniku. Za ovaj sustav nužan je ***dodjeljivač sabirnice***.

4.1 Koje računalne resurse dijele dretve istog procesa?

Dretve koje djeluju unutar jednog procesa dijele sva sredstva koja je operacijski sustav stavio na raspolaganje tom procesu.

4.2 Što je zajedničko dretvama različitih procesa?

Dva procesa ne mogu jedan drugome neposredno adresirati varijable već se razmjena podataka mora obaviti pomoću mehanizama OSa.

4.3 Kako je podjeljen spremnički prostor procesa, a kako dretveni spremnički pod-prostor?

PROCES dio za dretvene prostore
zajednički prostor (globalne varijable)

DRETVE dio za instrukcije
STOG
lokalni podaci

4.4 Navesti uvjet nezavisnosti zadataka.

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

4.5 Navesti uvjete koje mora zadovoljiti algoritam međusobnog isključivanja dretvi.

1) samo jedna dretva se smije nalaziti u kritičnom odsječku
2) algoritam mora djelovati uz proizvoljne brzine dretvi
3) ostanak neke dretve u nekritičnom odsječku ne smije drugoj dretvi onemogućiti ulazak u kritični dio
4) odabir dretve za kritični odsječak mora se obaviti u konačnom vremenu

4.6 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;
  }
}
```

Navedeni algoritam nije ispravan. Razlog – obje dretve mogu istovremeno ući u kritične odsječke tako da obje pročitaju u dva uzastopna sabirnička ciklusa vrijednosti 0, i na taj način obe uđu u kritični odsječak.

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

DEKKER ostvaruje međusobno isključivanje dviju dretvi
dvije zastavice ZASTAVICA[I, J]
varijabla PRAVO = {0,1}

LAMPORT ostvaruje međusobno isključivanje višeg broja dretvi
redni broj dolaska BROJ[N]
varijabla brojač ZADNJI_BROJ
niz ULAZ[N]

4.8 Navesti Dekkerov, odnosno Lamportov algoritam.

DEKKER

```
Dok je (1) {
    ZASTAVICA[I] = 1;
    Čitati varijablu ZASTAVICA[J];
    Dok je (ZASTAVICA[J] != 0) {
        Čitati varijablu PRAVO;
        Ako je (PRAVO != I) {
            ZASTAVICA[I] = 0;
            Dok je (PRAVO != I) {
                Čitati varijablu PRAVO;
            }
        }
        Čitati varijablu ZASTAVICA[J];
    }
    Kritični odsječak;
    PRAVO = J;
    ZASTAVICA[I] = 0;
    Nekritični odsječak;
}
```

LAMPORT

```
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;
}
```

4.9 Usporediti Petersonov i Dekkerov algoritam.

DEKKER ovisi o početnoj vrijednosti varijable PRAVO

PETERSON daje prednost bržoj dretvi
ne ovisi o početnoj vrijednosti varijable PRAVO

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

Najjednostavniji način međusobnog isključivanja na jednoprocesorskom računalu je ***zabranom prekidanja***.

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

TAS, SWAP, Fetch_And_Add...

4.12 U pseudokodu riješiti problem međusobnog isključivanja više dretvi uz pomoć nedjeljivih instrukcija TAS, SWAP i FETCH_AND_ADD. Koja je prednost tih rješenja u odnosu na Lamportov algoritam međusobnog iskljičivanja?

```
Dok je (1) {
    ˝Ispitati i Postaviti˝ varijablu ZASTAVICA;
    Dok je (ZASTAVICA != 0) {
        ˝Ispitati i Postaviti˝ varijablu ZASTAVICA;
    }
    Kritični odsječak;
    ZASTAVICA = 0;
    Nekritični odsječak;
}
```

Jednostavnost i pouzdanost rješenja u odnosu na Lamporta.

4.13 Koji je najveći zajednički nedostatak algoritmima međusobnog isključivanja (Dekkerov, Petersonov, Lamportov te algoritmima ostvarenima uz pomoć sklopovske potpore)?

Najveći nedostatak je taj što dretve izvode radno čekanje i time troše radne cikluse procesora na beskoristan posao.

5.1 Što predstavlja pojam ˝ulazak u jezgru˝ i kada se zbiva?

Poziv jezgrine funkcije ˝ulazak u jezgru˝zbiva se prekidom. Potrebno je pohraniti kontekst u opisnik dretve.

5.2 Na što se svodi ˝izlazak iz jezgre˝?

Izlazak iz jezgre svodi se na pokretanje jedne od dretvi, pri čemu se procesor vraća u korisnički način rada.

5.3 Navesti izvore prekida u jednostavnom modelu jezgre.

- programski prekidi koje izazivaju dretve
- sklopovske prekide od sata (periodično)
- sklopovske prekide od ulazno – izlaznih naprava

5.4 Od čega se sastoji jezgra operacijskog sustava?

Jezgra OSa sastoji se od strukture podataka jezgre i jezgrinih funkcija (smještene u sustavski dio spremnika).

5.5 Navesti sadržaj opisnika dretve.

- identifikator procesa
- identifikator dretve
- stanje dretve
- početna adresa dretvenog adresnog prostora
- veličina prostora
- adresa prve instrukcije
- zadano kašnjenje
- prostor za smještanje konteksta
- kazaljke

5.6 Navesti strukture podataka jezgre.

Strukturu podataka jezgre čine ***liste (redovi) dretvi***:

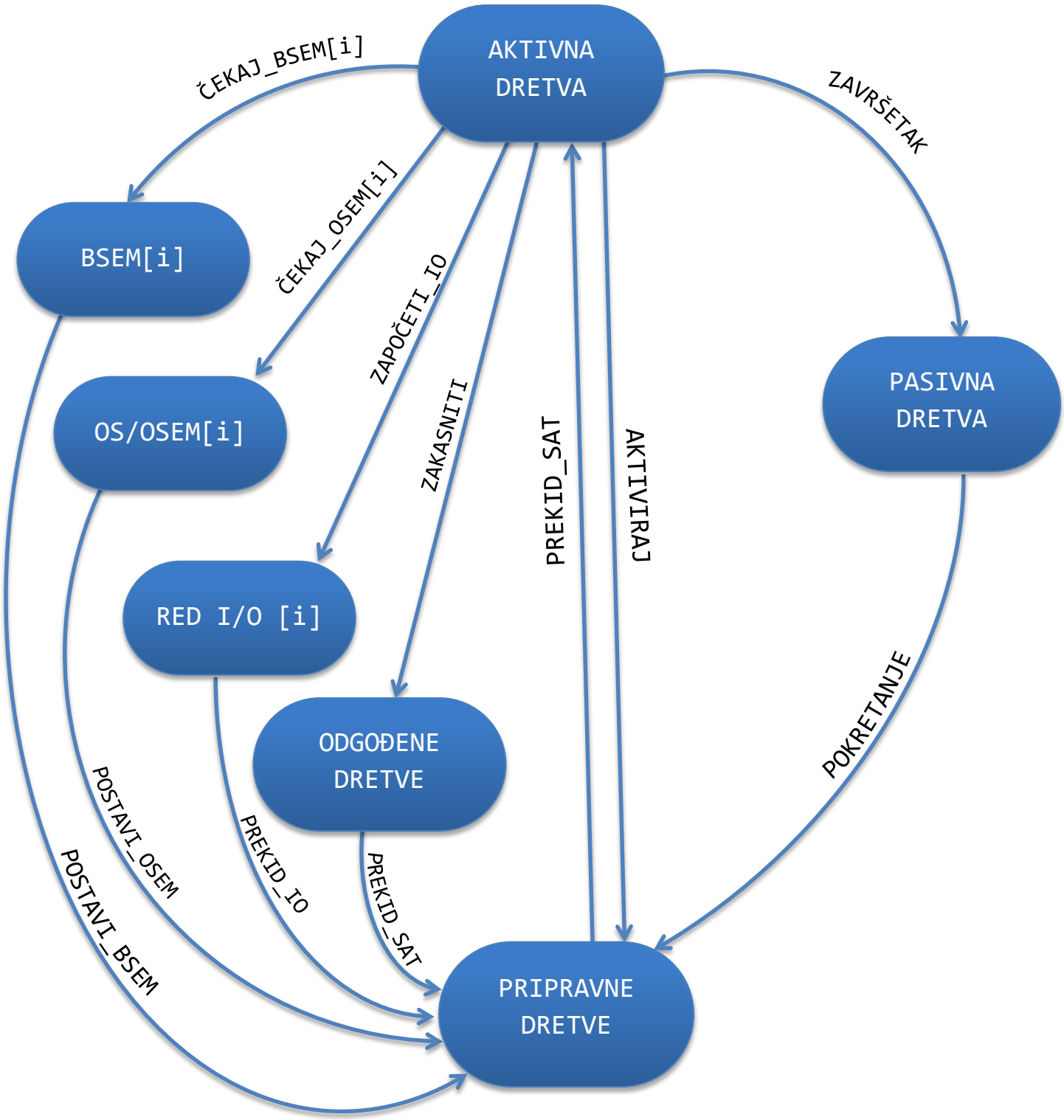
- Aktivne_Dretve
- Pasivne_Dretve
- Odgođene_Dretve
- Pripravne_Dretve
- Latentna_Dretva

i ***opisnik dretve***.

5.7 Koja su blokirana stanja dretvi?

- binarni semafor (BSEM)
- opći semafor (OSEM)
- odgođeno stanje (čekanje na istek vremena)
- čekanje na I/O operaciju

5.8 Skicirati graf mogućih stanja dretvi.



5.9 Što obavlja instrukcija ″AKTIVIRATI_PRVU_DRETVU_IZ_REDA_PRIPRAVNE_DRETVE″ ?

1) premjestiti prvi opisnik dretve iz reda PRIPRAVNE_DRETVE u red AKTIVNE_DRETVE
2) obnoviti kontekst iz opisnika aktivne dretve
3) omogućiti prekidanje
4) vratiti se u prekinutu dretvu

5.10 Što obavlja instrukcija ″VRATITI_SE_IZ_PREKIDNOG_NAČINA″?

Instrukcija ″**VRATITI_SE_IZ_PREKIDNOG_NAČINA**″ vraća u procesor sadržaj PCa i prevodi procesor iz sustavskog u korisnički način rada.

5.11 Čemu služe jezgrini mehanizmi binarni i opći semafor (BSEM i OS)?

Navedeni mehanizmi služe za sinkronizaciju dretvi.

5.12 Koje strukture podataka koriste BSEM, OS i OSEM?

BSEM zastavica `BSEM[i].v = {0,1}`
red `Red_BSEM`

OS varijabla `OS[i].v = {bilo koji cijeli broj}`
red `Red_OS`

BSEM varijabla `OSEM[i].v = {nenegativan cijeli broj}`
red `Red_OSEM`

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

```
jezgrina_funkcija Čekaj_BSEM(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Ako je (BSEM[I].v == 1) {
        BSEM[I].v = 0;
        Obnoviti kontekst iz opisnika Aktivna_Dretva;
        Omogućiti prekidanje;
        Vratiti se iz prekidnog načina;
    }
    Inače {
        Premjesti opsnik iz reda Aktivna_Dretva u
            red BSEM[I];
        Aktivirati prvu dretvu iz reda Pripravne_Dretve;
    }
}

jezgrina_funkcija Postavi_BSEM(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Premjesti opsnik iz reda Aktivna_Dretva u
            red Pripravne_Dretve;
    Ako je (red BSEM[I] neprazan) {
        Premjestiti prvi opisnik iz reda BSEM[I] u
            red Pripravne_Dretve;
    }
    Inače {
        BSEM[I].v = 1;
    }
    Aktivirati prvu dretvu iz reda Pripravne_Dretve;
}
```

```
jezgrina_funkcija Čekaj_OS(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    OS[I].v = OS[I].v – 1;
    Ako je (OS[I].v >= 0) {
        Obnoviti kontekst iz opisnika Aktivna_Dretva;
        Omogućiti prekidanje;
        Vratiti se iz prekidnog načina;
    }
    Inače {
        Premjesti opisnik iz reda Aktivna_Dretva u
            red OS[I];
        Aktivirati prvu dretvu iz reda Pripravne_Dretve;
    }
}

jezgrina_funkcija Postavi_OS(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Premjesti opsnik iz reda Aktivna_Dretva u
            red Pripravne_Dretve;
    OS[I].v = OS[I].v + 1;
    Ako je (OS[J].v == 0 && red OS[I] neprazan) {
        Premjestiti prvi opisnik iz reda OS[I] u
            red Pripravne_Dretve;
    }
    Aktivirati prvu dretvu iz reda Pripravne_Dretve;
}
```

```
jezgrina_funkcija Čekaj_OSEM(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Ako je (OSEM[I].v >= 1) {
        OSEM[I].v = OSEM[I].v - 1;
        Obnoviti kontekst iz opisnika Aktivna_Dretva;
        Omogućiti prekidanje;
        Vratiti se iz prekidnog načina;
    }
    Inače {
        Premjesti opisnik iz reda Aktivna_Dretva u
            red OSEM[I];
        Aktivirati prvu dretvu iz reda Pripravne_Dretve;
    }
}

jezgrina_funkcija Postavi_OSEM(I) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Premjesti opisnik iz reda Aktivna_Dretva u
            red Pripravne_Dretve;
    Ako je (red OSEM[I] neprazan) {
        Premjestiti prvi opsinik iz reda OSEM[I] u
            red Pripravne_Dretve;
    }
    Inače {
        OSEM[I].v = OSEM[I].v + 1;
    }
    Aktivirati prvu dretvu iz reda Pripravne_Dretve;
}
```

5.14 Opisati način umetanja opisnika dretve u listu ZAKAŠNJELE_DRETVE. Koja se vrijednost upisuje u polje ZADANO_KAŠNJENJE u opisniku dretve?

```
jezgrina_funkcija Zakasniti(L, M) {
    Pohraniti kontekst u opisnik Aktivna_Dretva;
    Premjesti opisnik iz reda Aktivna_Dretva u
            red Pripravne_Dretve;
    Pronaći opisnik dretve L u listi Postojeće_Dretve;
    Ako je (Dretva L nije pasivna) {
        Dojaviti grešku;
    }
    Inače {
        Premjesti opisnik dretve L u red Odgođene_Dretve;
    }
    Aktivirati prvu dretvu iz reda Pripravne_Dretve;
}
```

U polje ZADANO_KAŠNJENJE, odnosno vrijednost M, upisuje se vrijeme odgode dretve (broj perioda otkucaja sata).

5.15 Koje vrste prekida uzrokuju jezgrine funkcije ZAPOČETI_UI i PREKID_UI u jednostavnom modelu jezgre?

ZAPOČETI_UI programski prekid
PREKID_UI sklopovski prekid

5.16 Može li se prekinuti dretva koja obavlja neku jezgrinu funkciju ?

Dretvu koja obavlja jezgrinu funkciju ***nije moguće prekinuti***.

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

JEDNOPROCESORSKI SUSTAV ostvareno zabranom prekidanja
VIŠEPROCESORSKI SUSTAV sklopovska podrška / programski

6.1 Sinkronizirati proizvođača i potrošača korištenjem brojačkog semafora.

PROIZVOĐAČ

```
Dok je (1) {
    Proizvesti P;
    Međuspremnik[ULAZ] = P;
    ULAZ++;
    Postavi_OSEM(1);
 }
```

POTROŠAČ

```
Dok je (1) {
    Čekaj_OSEM(1);
    R = Međuspremnik[IZLAZ];
    Potrošiti R;
}
```

6.2 Sinkronizirati više proizvođača i više potrošača uz pomoć binarnih i brojačkih semafora.

PROIZVOĐAČ

```
Dok je (1) {
    Proizvesti P;
    Čekaj_OSEM(S);
    Čekaj_BSEM(K);
    Međuspremnik[ULAZ] = P;
    ULAZ++;
    Postavi_BSEM(K);
    Postavi_OSEM(I);
}
```

POTROŠAČ

```
Dok je (1) {
    Čekaj_OSEM(I);
    Čekaj_BSEM(K);
    R = Međuspremnik[IZLAZ];
    IZLAZ = (IZLAZ++) % N;
    Postavi_BSEM(K);
    Postavi_OSEM(S);
    Potrošiti R;
}
```

6.3 Sinkronizirati rad dviju dretvi tako da se one obavljaju naizmjenično.

PROIZVOĐAČ

```
Dok je (1) {
    Proizvesti P;
    Čekaj_OSEM(2);
    Međuspremnik = P;
    Postavi_OSEM(1);
}
```

POTROŠAČ

```
Dok je (1) {
    Čekaj_OSEM(1);
    R = Međuspremnik;
    Postavi_OSEM(2);
    Potrošiti R;
}
```

6.4 Što je potpuni zastoj?

Potpuni zastoj nastaje kada su dretve blokirane zauvijek (dretve se ne mogu nikako deblokirati). Kada bi jedna od dretvi uspjela krenuti ona bi deblokirala drugu.

6.5 Navesti nužne uvjete za nastajanje potpunog zastoja.

- neko sredstvo u istome trenutku može upotrebljavati samo jedna od dretvi (međusobno isključivo)
- dretvi se sredstvo ne može oduzeti (ona ga otpušta kada ga više ne treba)
- dretva drži dodjeljeno joj sredstvo dok čeka na dodjelu dodatnog sredstva