OPERACIJSKI SUSTAVI

PITANJA ZA VJEŽBU – 1. CIKLUS

(by kate)

1. Uvod

1. Što je Operacijski sustav?

Skup programa koji omogudavaju 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).

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

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 jedinice 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;
povečaj sadržaj programskog brojila tako da pokazuje na sljededu 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 sljededu instrukciju;
Ako je (dekodirana instrukcija poziv potprograma) {
Pohraniti sadržaj programskog brojila na stog;
Smanjiti sadržaj registra SP, tako da pokazuje na sljedede 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 omoguduju 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.

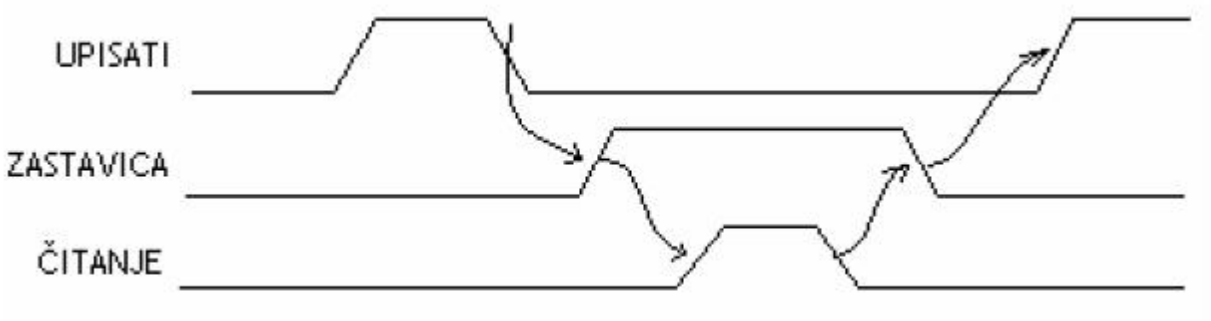
3. Obavljanje UI operacija, prekidni rad

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 taj događaj 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 omogućuje 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. omogudi 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;
Prijeđi u korisnički način rada;
Pozovi potprogram za obradu prekida j;

Zabrani prekidanje;
Prijeđi u sustavski način rada;
Vrati sadržaj iz KON[j] na sustavski stog i u varijablu T_P;
}
}
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 ved "uništen") ved to mora učiniti procesor. Također, vradanje 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 (3-52).

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 odgovarajudem 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 vedi 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 pomaknuda 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 koji samo on može pristupiti. Osim toga, svaki procesor može pristupiti do jednog zajedničkog djeljenog spremnika preko zajedničke sabirnice. U jednom sabirničkom ciklusu do spremnika može pristupiti samo jedan od procesora. Da bi odredili koji procesor može pristupiti spremniku imamo posebni sklopovski dodjeljivać sabirnice. Procesor I koji žeil pristup do djeljenog spremnika postavlja signal trazenja sabirnice T[I] dodjeljivacu sabirnice. Dodjeljivać na početku svog spremničkog ciklusa odlučuje kojem će procesoru dodjeliti sabirnicu. U jednom spremničkom ciklusu dodjeljivać će samo jednom procesoru dodjeliti sabirnicu, odnosno dodjeliti mu D[I]. Ako neki procesor postavi svoj zahtjev za dodjelu sabirnice on će u svom izvođenju zastati dok mu se sabirnica ne dodjeli. Dodjeljivac sabirnice dodjeljuje sabirnicu ciklički.

4. Međusobno isključivanje u višedretvenim sustavima

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

Roditejlu i djetetu nista nije zajedničko. 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 dohvadati 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 de udi u K.O. mora se zbiti u konačnom vremenu.

35. Za zadani algoritam medusobnog isključivanja 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 će isto biti u K.O. jer će Di ući 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 također nece uci u njega.

36. Čemu služi Dekkerov, a čemu 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. (04-51) i (04-63)

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 želi uci u K.O. ona zabrani prekide, i na izlasku iz K.O. ih ponovno omogući.

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 uvedan 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 udi 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 općem semaforu
6. **Red 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“?

Vrača 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. (05-41), (05-42), (05-60), (05-61), (05-66), (05-67)

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?

Zapoceti_UI je programski prekid, a Prekid_UI je sklopovski.

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 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.

6. Međudretvena komunikacija i koncepcija monitora

60. Sinkronizirati proizvodaca i potrošaca korištenjem brojackog semafora.

Proizvođač:

```
dok je(1){
        proizvesti poruku P;
        Ispitati_Osem(2);
        MS[UL]=P;
        UL=(UL+1)%N;
        Postavi_Osem(1);
}
```

Potrošač:

```
dok je(1){
        Ispitati_Osem(1);
        R=MS[IZ];
        IZ=(IZ+1)%N;
        Postavi_Osem(2);
        Potrošiti poruku R;
}
```

Objašnjene varijabli:
MS=međuspremnik
UL, IZ=kazaljke za kretanje po međuspremniku
P, R=poruka
N=veličina međuspremnika

61. Sinkronizirati više proizvođaća i više potrošaca uz pomoć binarnih i brojackih semafora.

Proizvođač:

```
Dok je (1){
        Proizvesti poruku P;
        Ispitati_Osem(S);
        Ispitati_Bsem(P);
        Dobaviti pretinac sa stoga SKLADISTE;
        Postaviti P u pretinac i uvrstiti ga u
        Red_Poruka[i];
        Postaviti_Bsem[P];
        Postaviti_Osem(i);
}
```

Potrošač:

```
Dok je (1){
        Ispitati_Osem(i);
        Ispitati_Bsem(K);
        Preuzeti poruku R iz prvog spremnika u
        Red_Poruka[i];
        Vratiti ispažnjeni pretinac na stog SKLADISTE;
        Postaviti_Bsem[K];
        Postaviti_Osem(S);
        Potrošiti poruku R;
}
```

62. Sinkronizirati rad dviju dretvi tako da se one obavljaju naizmjenicno.

Dretva Di:

```
dok je (1){
        Ispitati_Osem (i);
        nešto raditi;
        Postaviti_Osem(j);
}
```

Dretva Dj:

```
dok je (1){
        Ispitati_Osem (j);
        nešto raditi;
        Postaviti_Osem(i);
}
```

jedan OSEM je inicijalno postavljen na 0, a drugi na 1; Dretva u kojoj je OSEM postavljen u 1 ce prva poceti izvođenje.

63. Što je potpuni zastoj?

Potpuni zastoj je stanje sustava u kojem su sve dretve blokirane u nekom redu uvjeta.

64. Navesti nužne uvjete za nastajanje potpunog zastoja.

- sredstva se koriste međusobno isključivo (neko sredstvno u istom času može upotrebljavati samo jedna od dretvi)
- dretvi se sredstvo ne može oduzeti – ona ga otpusta sama kada ga vise ne treba
- dok traži dodatna sredstva dretva drži dodijeljena
- barem 2 dretve se natječu za barem 2 sredstva

65. Što je monitor?

Monitor je jezgrin mehanizam za sinkronizaciju.

66. Navesti jezgrine strukture podataka koje se koriste za ostvarenje monitora.

- monitorski semafor
- redovi uvijeta

67. Navesti jezgrine funkcije za ostvarenje monitora.

- ući_u_monitor(M),
- izaći_iz_monitora(M),
- uvrstiti_u_red_uvjeta(M,K),
- osloboditi_iz_reda_uvjeta(M,K)

68. Kada se može dogoditi da se dvije dretve nadu u monitoru?

Jedna dretva je upravo deblokirana iz reda uvjeta, a druga koja je tu dretvu deblokirala, a još nije završila izvođenje i izašla iz monitora.

69. U pseudokodu napisati jezgrine funkcije za ostvarenje monitora: Uci_u_monitor, Izaci_iz_monitora, Uvrstiti_u_red_uvjeta i Osloboditi_iz_reda_uvjeta.

```
Uci_u_monitor(M){
        Pohraniti kontekst u opisnik Aktivna_D;
        Ako je (Monitor[M].v==1) {
                Monitor[M].v=0;
                Obnoviti kontekst iz opisnika Aktivna_D;
                Omogućiti prekidanje;
                Vratiti se iz prekidnog načina rada;
        }
        Inače {
                Premjestiti opisnik iz reda Aktivna_D u red Monitor[M];
                Aktivirati prvu dretvu iz reda Pripravne_D;
        }
}

Izaci_iz_monitora(M){
        Pohraniti kontekst u opisnik Aktivna_D;
        Premjestiti opisnik iz reda Aktivna_D u red Pripravne_D;
        Ako je ((Monitor[M].v==0) && (red Monitor[M] neprazan)) {
                Premjesti prvi opisnik iz reda Monitor[M] u red Pripravne_D;
```

```
        }
        Inače
                Monitor[M].v=1;
        Aktivirati prvu dretvu iz reda Pripravne_D;
}

Uvrstiti_u_red_uvjeta(M,K){
        Pohraniti kontekst u opisnik Aktivna_D;
        Premjestiti opisnik iz reda Aktivna_D u Red_uvjeta[M,K];
        Ako je ((Monitor[M].v==0) && (red Monitor[M] neprazan)) {
                Premjesti prvi opisnik iz reda Monitor[M] u red Pripravne_D;
        }
        Inače
                Monitor[M].v=1;
        Aktivirati prvu dretvu iz reda Pripravne_D;
}

Osloboditi_iz_reda_uvjeta(M,K){
        Pohraniti kontekst u opisnik Aktivna_D;
        Premjestiti opisnik iz reda Aktivna_D u red Pripravne_D;
        Ako je (Red_uvjeta[M,K] neprazan) {
                Premjesti prvi opisnik iz reda Red_uvjeta[M,K] u red Pripravne_D;
        }
        Ako je ((Monitor[M].v==0) && (red Monitor[M] neprazan)) {
                Premjesti prvi opisnik iz reda Monitor[M] u red Pripravne_D;
        }
        Inače
                Monitor[M].v=1;
        Aktivirati prvu dretvu iz reda Pripravne_D;
}
```

70. Kojim jezgrinim mehanizmom moraju biti zašticene korisnicke monitorske funkcije?

Monitorskim semaforima

71. U cemu se razlikuje monitorski semafor od binarnog semafora?

Binarni semafor koristi samo dvije funkcije, a monitorski četiri

7. Analiza vremenskih svojstava racunalnih sustava.

72. Objasniti parametre sustava i njihov odnos:

ρ - iskoristivost $\rho = \alpha/\beta$

α - broj poslova koji u jedinicnom vremenu ulaze u sustav

$1/\alpha$ – prosjecno vrijeme izmedju 2 posla

β -broj poslova koje poslužitelj može obaviti u jedinici vremena

$1/\beta$ – prosjecno trajanje posluživanja

n - prosjecni broj poslova u sustavu $n = \alpha\ T = \rho/(1-\rho) = \alpha/(\beta- \alpha)$

T -prosjecno zadržavanje posla u sustavu $T = 1/(\beta - \alpha)$

$p_i = (1-\rho)\rho^i$ - vjerojatnost da u sustavu imamo i poslova
$p(i>N) = \rho^{N+1}$

$\alpha = \alpha_1 + \alpha_2 + \ldots$
$\rho = \rho_1 + \rho_2 + \ldots$
$1/\beta = \alpha_1/\alpha * 1/\beta_1 + \alpha_2/\alpha * 1/\beta_2 + \ldots$

73. Navesti Littleovo pravilo. $n = \alpha T$

74. Skicirati Markovljev lanac gdje stanja predstavljaju broj poslova u sustavu. Sustav neka ima Poissonovu razdiobu dolazaka s parametrom α i eksponencijalnu razdiobu trajanja obrade s parametrom $1/\beta$. (07-49)

76. U kojem slucaju se može dozvoliti veliki faktor iskorištenja (cak i 1)?

U deterministčkom slučaju. U stohastičkom slučaju faktor iskorištenja je strogo manji od 1 (<1), dok je u deterministčkom slučaju <=1

8. Gospodarenje spremnickim prostorom

78. Gdje se generiraju adrese unutar procesora?

Programsko brojilo → adrese instrukcija
Registar kazaljke stoga → stogovne adrese
Sadržaj adresnih dijelova instrukcija → adrese podataka

79. Kako je podijeljen procesni adresni prostor?

Dretveni prostori + zajednički prostor

80. Opisati organizaciju smještaja sadržaja na magnetskom disku (cilindri, staze, sektori).

Disk se sastoji od više ploča na istoj osovini. Kružnice na ploči predstavljaju staze, a staze su podjeljene na jednake djelove (kružne lukove s jednakim središnjim kutem) – sektore. Staze jednakih polumjera svih diskova čine cilindar.

81. Čime je odredena jedinstvena adresa svakog sektora na disku?

- Rednim brojem ploče
- Rednim brojem staze na ploči
- Rednim brojem sektora na stazi

83. Od cega se sastoji trajanje traženja staze (*seek time*)?

- Ubrzavanje (ručice glave)
- Gibanje konstantnom brzinom
- Usporavanje
- Fino pozicioniranje

84. Koliko iznosi prosjecno vrijeme traženja u odnosu na vrijeme koje je potrebno za prijelaz preko svih staza?

Za vrijeme trazenja se uzima trečina vremena potrebnoh za prijelaz svih staza

82. Koliko iznosi ukupno trajanje prijenosa podataka tvrdi disk - radni spremnik? Pogledaj 85.

85. Kako je podijeljeno vrijeme koje je potrebno za prijenos podataka s diska ili na disk?

1) Trajanje postavljanja glave (head positioning time)
    - Trajanje traženja staze (seek time)
        - Ubrzavanje ručice glave
        - Gibanje konstantnom brzinom
        - usporavanje
        - fino pozicioniranje
    - rotacijsko kašnjenje ($\overline{T_r} = T_r/2$)
2) Trajanje prijenosa podataka (data transfer time)
    - Trajanje čitanja dijela ili cijele staze

86. Zbog čega nastaje rotacijsko kašnjenje i koliko ono iznosi?

Rotacijsko kašnjenje (T_r) nastaje zbog toga što se nakon postavljanja glave na odabranu stazu mora prije početka prijenosa pričekati da se ispod glave za čitanje pojavi traženi sektor.
Iznosi između nula(najpovoljniji slučaj traženi sektor upravo nailazi ispod glave u trenutku njezinog postavljanja na stazu) i jednog (najlošiji slučaj traženi sektor upravo prolazi ispod glave kad je ona postavljena na stazu) okretaja. Zato se za rotacijsko kašnjenje uzima prosječna vrijednost, odnosno pola trajanja jednog okretaja (T_r=T_r/2).

87. Čime je odredena brzina prijenosa podataka s diska u spremnik diskovne upravljacke jedinke?

Brzinom kojom ispod glave promiču bajtovi sektora i brojem podataka koje trebamo prenjeti.

88. Neka je trajanje traženja staze *Ts* nekog diska s 2000 staza opisano sa sljedece tri formule, gdje je *D* udaljenost izmedu trenutnog položaja glave i tražene staze:

- *Ts = 1.5 x D ms za D ≤ 4,*
- *Ts = 4.0 + 0.5 x $D^{1/2}$ ms za 4<D ≤ 400,*
- *Ts = 10.0 + 0.01 x D ms za D > 400.*

Koliko iznosi prosjecno vrijeme traženja staze?

$$T_s = 10.0 + 0.01 * \frac{2000}{3} = 16.66\ ms$$

89. Navesti sadržaj procesnog informacijskog bloka.

Tablice sektora u kojima je program procesa pohranjen na disku i podaci o smještaju programa u radnom spremniku.

90. Opisati postupke statickog i dinamickog dodjeljivanja spremnika.

Statičko dodjeljivanje:

Particije su stalne veličine (fixed partitions), program kad je dodjeljen jednoj particiji uvijek je u toj particiji, i kad je izbačen iz radnog spremnika može se vratiti samo u tu particiju. Dolazi do problema fragmentacije i toga što adresni prostor programa nije mogao biti veći od najveće particije fizičkog spremnika.

Dinamičko dodjeljivanje:

Programi se upisuju jedan iza drugog. Dodan je sklop s kojim se početna adresa pribraja svakoj adresi koju proces proizvede. Programi se priređuju tako da svaki ima svoj adresni prostor. Važno je da se prije izvođenja u bazni registar pohrani aktualna početna adresa.

91. Navesti vrste fragmentacije prilikom statickog dodjeljivanja spremnika.

Unutarnja fragmentacija: Programi nisu jednake velicine kao particije pa ce djelovi particija ostati neiskorišteni.

Vanjska fragmentacija: Tijekom rada se može dogoditi da svi procesi čiji su programi smješteni u istu particiju bivaju blokirani pa ta particija radnog spremnika ostaje prazna. Pritom može postojati više procesa čiji programi čekaju na dodjelu radnog spremnika, ali oni ne mogu biti napunjeni u radni spremnik jer nisu pripremljeni za tu particiju.

92. Problem fragmentacije prilikom dinamickog dodjeljivanja spremnika se ne može izbjeci, ali se može ublažiti. Kako?

Rupe se održavaju što večima kako bi se u njih mogo smjestiti novi program.

- Pri svakom oslobađanju nekog procesnog prostora novo nastala rupa spaja se s eventualnim susjednim rupama u novu veću rupu
- Pri svakom novom zahtjevu za spremničkim prostorom potraži se najmanja rupa u koju se može smjestiti novi program

93. Unatoč tome što se problem fragmentacije prilikom dinamickog dodjeljivanja spremnika može ublažiti, fragmentacija može postati prevelika. Što treba tada učiniti?

Privremeno obustaviti izvođenje dretvi i „presložiti“ programe u kompaktni prostor.

94. Navesti i dokazati Knuthovo 50% pravilo.

Knuthovo pravilo: u stacionarnom stanju će u spremniku biti broj rupa jednak polovici broja punih blokova.

Izvod:

Knuthovo pravilo pretpostavlja da je vjerojatnost oslobađaja spremničkog prostora jednaka vjerojatnosti zahtjeva. Pretpostavlja se da će zbog toga tijekom vremena u sustavu doći do stohastičkog ravnotežnog stanja odnosno da će prosječan broj rupa i

punih blokova biti stalan. U tom slučaju možemo predpostaviti da postoje 4 tipa blokova razvrstavjući ih po kriteriju da li im im jepredhodnik i sljedbenik rupa ili puni blok.
A xxx, B xx0, C 0xx, D 0x0
Veliko slovo je tip bloka, a malo slovo je broj blokova tog tipa. Blokova tipa A ima a.

Vjerojatnost da će se pronaći neka rupa koja će točno odgovarati veličini zahtjevanog prostora je zanemariva i označava se sa *q*, a vjerojatnost (*p= 1 - q*) je zato približno jednaka 1.

Sad idemo prebrojat rupe, blokove tipa ne brojimo jer oni nemaju rupa, a blokove tipa D brojimo dva puta jer imaju po dve rupe. E sad budući da ako, recimo, imamo blok tipa B(xx0) i
blok tipa C(0xx) jedan do drugoga vidimo da smo ovu nulu brojili dva puta
i zato moramo cijelu sumu podjelit još sa dva.
n(rupa) = (b+c+2d)/2

Uz to budući da ako jednom bloku (C) slijedi nula, to mora značiti da će slijedećem (B)
prethoditi, što znači da će broj blokova tipa C i tipa B bit jednak **c=b** .

Znači,
n(rupa)=(2b+2d)/2 = b + d
m(količina punih segmenata)=a + b + c + d

Vjerojatnost da će nastati jedna rupa (znači mora nestat jedan blok tipa A) je **a/m**.
Vjerojatnost da će se popunit rupa (mora nestat blok tipa D) je **d/m + vjerojatnost** da postoji zahtjev veličine jedne r upe u koju će se novi program smjestiti (ili kraće q).
Zbog onog prvog uvjeta vjerojatnost popunjavanja i pražnjenja rupa moraju biti otprilike jednaki znači:
a/m = d/m + q ; pomnoži s m, zamjeni q sa 1-p
a=d + m(1-p)

uvrstimo sad taj a u
m=a+b+c+d;
m=d + m(1-p) +b+c+d ;

sad se sjetimo odprije da je c=b i b+d=n
m=d + m-pm + b +b + d
m=m-pm +2(n)
pm = 2n
n=1/2pm ; udući da je p približno 1
n(rupa)=1/2m(memorije)

50% memorije čine rupe .

95. Kako treba podijeliti program (koji u cijelosti ne stane u radni spremnik) u prekloponom nacinu uporabe radnog spremnika?

Na jedan osnovni dio koji se uvijek nalazi u radnom spremniku i na dijelove koji se naizmjenično smještaju u preostali dio korisničkog dijela fizičkog spremnika.

96. Kako je podijeljen logicki, a kako fizicki adresni prostor u sustavu sa stranicenjem?

Logički adresni prostor se dijeli na jednako velike dijelove koje nazivamo **stranicama** (*page*). Prikladno je da stranice imaju veličinu koja je cjelobrojna potencija broja dva.

Fizički adresni prostor možemo podijeliti na dijelove koji su jednaki veličini stranice logičkog adresnog prostora. Te zamišljene dijelove nazivamo **okvirima** (*frames*). U jedan okvir fizičkog radnog spremnika može se smjestiti jedna stranica logičkog adresnog prostora

97. O cemu ovisi velicina fizickog i logickog adresnog prostora?

Logički adresni prostor ovisi o arhitekturi, a fizički o RAM-u.

98. Mogu li stranice logickog adresnog prostora biti smještene u okvire fizickog spremnika proizvoljnim redoslijedom?

Mogu. Time se izbjegava problem fragmentacije.

99. Cemu služi tablica prevodenja? Od kojih se elemenata sastoji?

Služi za prevođenje logičkog u fizički adresni prostor. Sastoji se od direktorija stranica i tablica stranica.

Tablica prevođenja ima dvije razine adresa organizirane po stranicama veličine 4 KB. U prvoj razini je jedna stranica s 1024 kazaljki od po 32 bita (direktorij stranica). Kazaljke direktorija pokazuju na stranice u kojima je opet smješteno po 1024 riječi od po 32 bita (tablica stranica)

100. U cemu se razlikuju prekidi izazvani zbog promašaja stranice kod stranicenja na zahtjev od ostalih vrsta prekida?

Prekid zbog adresiranja nepostojeće stranice se dogodi usred instrukcije. Nakon obrade prekida mora se ponoviti instrukcija unutar koje se dogodio prekid. Kod ostalih vrsta prekida, postojanje prekida se provjerava nakon izvodjenja instrukcije, a ponovno pokretanje dretve započinje sljedečom instrukcijom.

101. Opisati sklopovsku potporu za ostvarenje stranicenja u Intel x86 arhitekturi. Što sadrži i cemu služi TLB (Translation Lookaside Buffer)? Opisati tablicu prevodenja. Gdje se ona nalazi?

Intel nema registra povijesti. Stranice se razvrstavaju u dva razreda: one u koje se u
prethodnoj periodi pisalo i one u koje se nije
pisalo. Postoji jedan bit pristupa(A) i bit nečistoće(D). Prvo se pokuša izbaciti stranica kojoj je A=0. Inače se izbacuje s tranica
kojoj je bit nečistoće 0, odnosno čista stranica.
TLB je 32 bitni priručni međuspremnik za prevođenje adresa koji nam služi za cjelokupno spremanje fizičke adrese i dijela logičke adres, tj. omogućuje nam da ne moramo uzastopno dohvaćati fizičku adresu ako ju trebamo češće.

Koliko puta je potrebno pristupiti radnom spremniku ako se stranica ne nalazi u TLB meduspremniku? 3 a koliko ako se nalazi u TLB medusprenmiku? 1

102. Koju informaciju nosi bit cistoce? Gdje se on nalazi?

Označava da li se sadržaj stranice mijenjao od njenog prebacivanja u radni spremnik. Ako se mijenjao kod izbacivanja iz radnog spremnika se mora prepisivati na disk, a ako nije onda se samo obriše iz radnog spremnika.

103. Čemu služi posmačni registar povijesti?

Za izvedbu LRU strategije. Pri punjenju stranice u neki odvir inicijaliziraju se na 0 njezin bit pristupa i posmačni registar. Kada se pristupi stranici bit pristupa stranice se postavlja u 1. Prekid od sata periodno posmakne sve bitove pristupa u posmačne registre i zatim bit pristipa izbriše. Sadržaj registra može se pročitati kao binarni broj. Manji broj pokazuje da se stranicu nije duže vrijeme koristilo. Kada se pojavi potreba za izbacivanjem stranice izbacuje se ona s najmanjim brojem.

104. Opisati sljedece strategije za izbacivanje stranica: FIFO, LRU, OPT te satni algoritam.

- FIFO – izbacuje se stranica koja je najdulje u radnom spremniku (first-in first-out)
- LRU – izbacuje se stranica koja se najdulje u proslosti nije upotrebljavala (least recently used)

- OPT (Optimalna strategija) – izbacuje se stranica koja se najdulje u budućnosti neće upotrebljavati
- Satni algoritam – Stranice se svrstavaju u listu po redu prispjeća. Lista se obilazi kružno posebnom kazaljkom (kada dođe do kraja kazaljka se vraća na početak liste). Kada se pojavi potreba za praznim okvirom izbacit će se stranica na koju pokazuje kazaljka ako je njezina zastavica A u 0, inače će se preskočiti i pritom promjeniti vrijednost A u 0. Kazaljka se pomiče za jedno mjesto dok ne naiđe na stranicu s A=0.

105. Opisati strategiju izbacivanja stranica u Intel x86 arhitekturi kada se u obzir uzimaju dvije zastavice za oznacavanje stanja stranice.

Zastavice koje se koriste su bit pristupa (A) i bit nečistoće (D). Prije se izbacuju stranice u kojima se nije pristupalo i koje su čiste.

106. U kojim stanjima se mogu nalaziti pojedini okviri tijekom rada?

- Aktivno stajne – okvir je dodjeljen jednom od procesa i njegov se redni broj (kazaljka koja pokazuje na njegov početak) nalazi u tablici prevođenja tog procesa
- Slobodno stanje – okvir se nalazi u povezanoj listi oslobođenih okvira
- Slobodno stanje s obrisanim sadržajem – okvir je spreman za dodjeljivanje

9. Datotečni podsustav

107. Navesti sadržaj opisnika datoteke.

- Naziv datoteke
- Tip
- Lozinka
- Ime vlasnika
- Prava pristupa
- Vrijeme stvaranja
- Vrijeme zadnje uporabe
- Ime posljednjeg korisnika, i slično
- Opis smještaja

108. Gdje su pohranjeni

a) opis smještaja datoteke - u opisniku datoteke
b) opisnik datoteke - na disku u nekom volume-u
c) datotecna tablica - u opisniku datoteke

109. Navesti sadržaj datotecne tablice.

- broj sektora po disku (kapacitet)
- broj slobodnih sektora (veličina slobodnog prostora)
- informacije o slobodnim sektorima
- tablica opisnika pohranjenih datoteka

110. Na koji nacin se može prikazati slobodan prostor na disku.

- bitovni prikaz
- lista slobodnih sektora
- lista nakupina sektora

111. Opisati bitovni prikaz slobodnog prostora na disku.

U nizu bitova svakom sektoru odnosno nakupini sektora pripada jedan bit čija je vrijednost jednaka 1 kada je sektor odnosno nakupina zauzeta, inace 0.

112. Opisati prikaz slobodnog prostora u obliku liste slobodnih blokova.

Svaki element liste sadrzi tri lokacije. Prvo je kazaljka na sljedeci slobodan element, drugo adresa prvog slobodnog sektora, a trece broj slobodnih sektora.

113. Opisati nacin smještaja datoteka u UNIX datotecnim podsustavima (*i-node*).

Opisnik datoteke u UNIXu zove se i-node. Uz pretpostavku da je sektor velik 1KB u jedan sektor se može smjestiti 256 kazaljki. U opisniku se nalazi 13 kazaljki i to:

- 10 neposrednih kazaljki
- 1 jednostruko indirektna kazaljka
- 1 dvostruko indirektna kazaljka
- 1 trostuko indirektna kazaljka

Pristup do datoteke obavlja se na sljedeći način:

- Prvih deset sektora dohvaća se neposrednim kazaljkama čime se može dohvatiti sve sektore datoteka manjih od 10KB
- Jedanaesta kazaljka pokazuje na sektor u kojem se nalazi sljedečih 256 kazaljki (256 + 10)KB
- Dvanaesta kazaljka pokazuje na sektor u kojem su kazaljke na 256 sektora svaki s 256 kazaljki (256*256+256+10)KB
- Trinaesta kazaljka pokazuje na trostuko stablo kazaljki (256*256*256+256*256+256+10)KB = 16GB

114. Opisati nacin smještaja datoteka u NTFS datotecnom podsustavu. Što je MFT? Kako se pohranjuju „male“ datoteke?

Diskovni prostor dodjeljuje se po skupinama sektora (po 1, 2, 4 ili 8 sektora). NTFS sadrži datoteku MFT (glavna tablica datoteka) u kojoj se nalaze opisnici svih datoteka (ukljucujuci i nje same). Jedan MFT zapis ima veličinu jedne skupine sektora (npr. 4KB). Za veče datoteke zapis sadrži indekse skupina sektora i ako je potrebno dodatno proširenje opisa.

Datoteka se dijeli na djelove koji su jednako veliki kao nakpine sektora (virtualne skupine). Za smještaj datoteka pronalazi se što više uzastopnih skupina sektora i dodjeljuje se datoteci (ako nema uzastopnih dodjeljuju se pojedinačne skupine).

Male datoteke smještaju se unutar MFT zapisa.