PITANJA ZA ZAVRŠNI ISPIT IZ OPERACIJSKIH SUSTAVA verzija 1.0

by Aeroflot

UVODNA NAPOMENA

NE ODGOVARAM ZA GREŠKE!!!!

Ako krivo naučite iz ove skripte, sami ste si krivi jer mi vjerujete XD.

I da… hvala svima koje sam pokrao. :D Sad ću bezobrazno stavit svoj copyright i dilat po fakultetu. XD

p.s. downloadom ove skripte automatski pristajete autoru donirati bubreg, jetru ili bilo koji drugi organ u slučaju potrebe :D

1. CIKLUS PREDAVANJA

1. i 2. POGLAVLJE

1. Što je OS? (OK) 01-02

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

2. Nabrojite osnovne dijelove OS-a! (OK), slika

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.

4. Čime su određena svojstva i ponašanje procesora? (OK) 02-15

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).

5. Nabroji osnovni skup registara procesora? (OK) 02-16

Adresni međuregistar	(za postavljanje adrese)
Podatkovni međuregistar	(za dohvat odnosno spremanje podataka)
Instrukcijski registar	(spremaju se insturkcije kad se dobave)
Programsko brojilo (PC)	(za dohvat instrukcija, pokazuje na sljedeću instrukciju)
Registar kazaljke stoga (SP)	(stack pointer)
Registar stanja (SR)	(registar zastavica)
Registri opće namjene (R)	

6. Što je kontekst dretve? (OK) 02-42

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

7. Definirajte program, proces i dretvu! (OK) 02-39

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.

8. Kako je moguć višeprogramski rad na jednom procesoru? (OK)

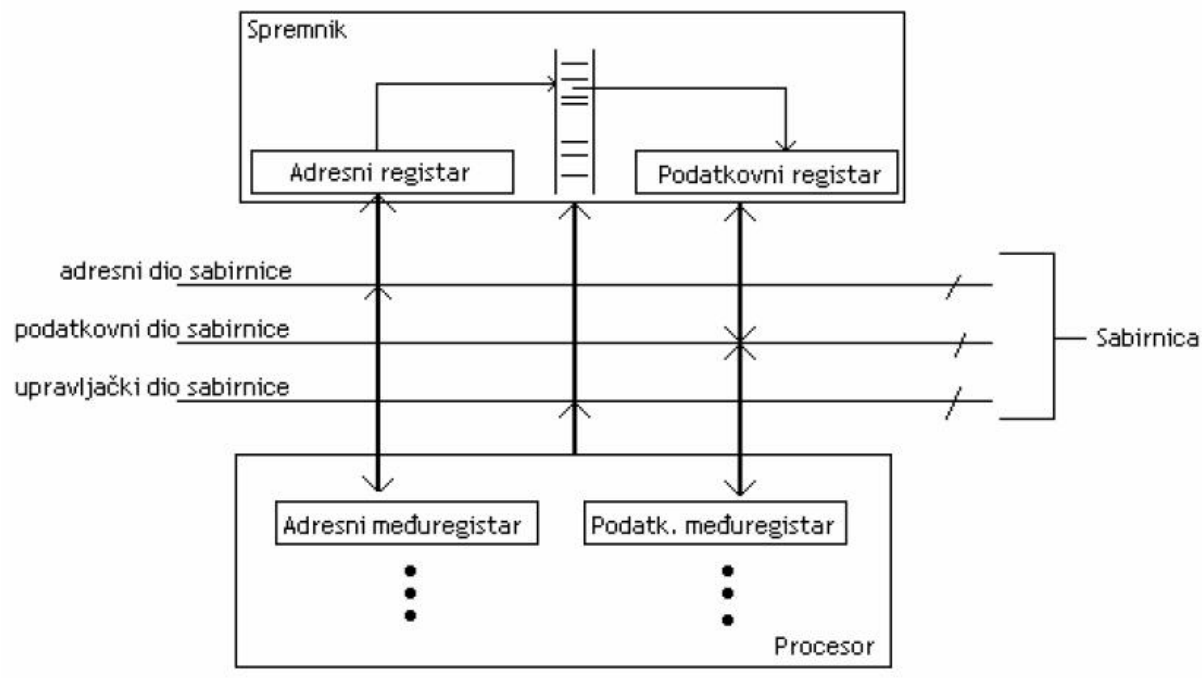
Tako da se svaka dretva izvodi naizmjence pa dobijemo privid istovremenosti. Ključna je pravila izmjena konteksta dretve.

9. Napisati pseudokod što procesor trajno radi! (OK) 02-29

```
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 pokhranjuje rezultat;
        operande dovesti na aritmetičko-logičku jedinku, izvesti zadanu operaciju;
        pohraniti rezultat u odredište;
} dok je (procesor uključen);
```

Procesor se može promatrati kao automat koji, nakon uključivanja, trajno izvodi instrukciju za instrukcijom strojnog programa.

10. Skicirati procesor i spremnik povezan na sabirnicu. Sabirnicu podijeliti na tri dijela! (OK) 02-12



11. Opisati osnovne komponente pristupnog sklopa. (OK) 03-06

- sklop za dekodiranje adrese (dobavlja adresu podatka)
- podatkovni registar (PR) (za podatke)
- registar stanja (RS) (daje procesoru stanje u kojem se nalazi naprava)
- upravljački dio (određuje stanje naprave)

12. Što je to adresni prostor?

Adresni prostor je dio spremnika potreban procesoru koji se dijeli na:

- dio u kojem je smješten program - segment instrukcija
- dio rezerviran za stog
- dio za podatke (korisnički i sustavski podaci)

3. POGLAVLJE

13. Što je radno čekanje? (OK) 03-06, 03-15

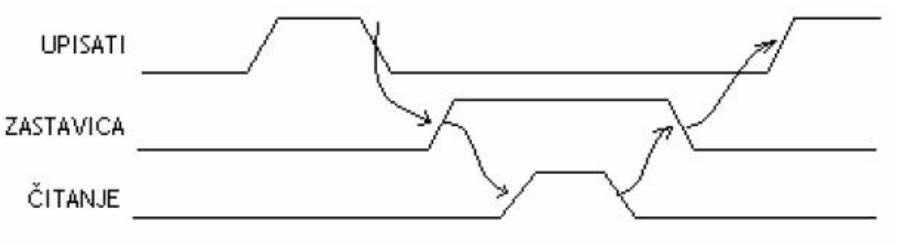
Režim rada procesora u kojem procesor čeka na određeni događaj u programu (npr. pojava zastavice) i troši vrijeme dok se on ne dogodi.

(primjer pseudokoda):

```
čitati registar sranja RS;
        dok je (ZASTAVICA ==0){
                čitati registar stanja RS;
        }
        čitati registar PR; //time se automatski briše ZASTAVICA
```

ako se na takav način čita niz znakova onda ovu petlju treba ponavljati u nekoj nadređenoj petlji

14. Skicirati signale dvožičnog rukovanja između pristupnog sklopa i U/I naprave! (OK) 03-16...



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

15. Što se zbiva kad se dogodi prekid? (OK) 03-22, 03-23

Pojava prekidnog signala prebacuje procesor u tzv. sustavski (jezgreni) način rada – pozivaju se podprogrami koji sačinjavaju jezgru OS-a. Također procesor čini sljedeće:

1. onemogućuje daljnje prekidanje
2. adresira sustavski dio spremnika (spremnik je time podijeljen na korisnički i sustavski)
3. aktivira sustavski registar kazaljke stoga koji se nalazi u procesoru
4. pohranjuje trenutni sadržaj programskog brojila (PC) na stog
5. u programsko brojilo stavlja adresu na kojoj počinje podprogram za obradu prekida
6. pohranjuje kontekst dretve na sustavski stog
7. nakon što je kontekst spremljen može se obaviti stvarna obrada prekida
8. nakon obrade pokreće prekinutu korisničku dretvu
9. sa sustavskog stoga vraća sadržaj registara
10. omogućuje prekidanje
11. prebacuje adresiranje u korisnički adresni prostor
12. aktivira korisnički registar kazaljke stoga
13. vraća sadržaj programskog brojila (čime se pokreće prekinuta dretva)

16. Objasniti instrukciju vratiti se iz prekidnog načina rada, obnoviti kontekst i pohraniti kontekst! (OK) 03-27...

a) vratiti se iz prekidnog načina rada

Prije no što se programsko brojilo sa sustavskog stoga vrati u PC (sadržaj) treba omogućiti prekidanje, prebaciti adresiranje u korisnički adresni prostor i aktivirati korisnički registar kazaljke stoga.

b) pohraniti kontekst

Sadržaj registara procesora prilikom izvođenja dretve naziva se kontekst dretve.

Kontekst dretve treba pri prekidanju izvođenja jedne dretve pohraniti izvan procesora (osim PC registra koji se tretira zajedno!) i u registre staviti kontekst dretve koju se želi pokrenuti.

Pohraniti kontekst znaci pohranjivanje sadržaja svih registara procesora na sustavski stog, radi se na početku podprograma.

c) obnoviti kontekst

Na kraju podprograma treba vratiti registre obrnutim redoslijedom.

17. Što radi potprogram za obradu prekida na početku? (OK) 03-26

Pohranjuje sadržaj svih registara procesora (kontekst) na sustavski stog.

18. Kako završiti s prekidom? (OK)

- obnoviti kontekst sa sustavskog stoga
- omogućiti prekidanje *(odgođeno do upisa nove vrijednosti u PC), prebacuje adresiranje u korisnički prostor*
- vratiti se iz prekidnog načina rada *(aktivira korisnički registar kazaljke stoga)*
- vratiti PC

19. Skicirati sklop za prihvat prekida! (VIDI) 03-51,52

20. Čemu služi i kako djeluje sklop za prihvat prekida? (OK) 03-50

Sklop za prihvat prekida služi za filtriranje prekida tako da kad se dogodi prekid manjeg prioriteta sklop ne šalje prekid procesoru („ne ometa ga“ tako da procesor ni nezna da je bio prekid). Odnosno prema procesoru se propušta samo prekid višeg prioriteta od dretve koju procesor trenutno izvodi.

21. Koje strukture podataka treba sadržavati OS koji omoguduje prihvat prekida različitih prioriteta? (OK)

Varijablu T_P (tekući prioritet), **polje KON[N]** (kontekst pojedine dretve) i **polje K_Z[N]** (kopije zastavica).

22. Što su programski prekidi i tko ih izaziva? (OK)

Programski prekidi su instrukcije pomoću kojih programi pozivaju jednu od jezgrinih funkcija. Može se javiti zbog greške u programu (dijeljenje s 0, nepostavljena varijabla) ili ih možemo postaviti u programu kada je potrebno kontekst (podatke u registrima procesora) obraditi u jezgrenom načinu rada.

23. Što sve uzrokuje prekid generiran unutar procesora? (OK) 03-60

- dijeljenje sa nulom
- adresiranje nepostojeće lokacije u adresnom prostoru dretve
- dekodiranje nepostojećeg instrukcijskog koda

24. Što obavlja instrukcija vratiti se iz prekidnog načina?

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

25. Zašto se PC tretira zasebno prilikom pohrane konteksta? (OK)
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 PC-a pokreće prekinutu dretvu!

26. Zašto u kontekstu nije uključeno programsko brojilo? (OK) – nešto slično ko 25
Zato jer se kod prekida prvo stavlja sadržaj programskog brojila na stog a zatim kontekst kako bi se pri vraćanju iz prekida mogao kontekst obnoviti prije nego se vrati stara vrijednost programskog brojila (što će pokrenuti dretvu).

27. Opisati DMA sklop i njegove registre! (OK) 03-64...
DMA služi prijenosu blokova znakova/podataka. Odvijanje prijenosa cijelog bloka između dvaju prekida obavlja se bez sudjelovanja procesora.
Registri pristupnog sklopa:

- 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)

28. Koje su razlike prijenosa znakova prekidnim mehanizmom i DMA prijenosom? (OK)
Kod prekidnog mehanizma javljati će se prekid koji će zahtjevati dodatnu količinu instrukcija za spremanje i obnovu konteksta i zato će se gubiti dio radnog vremena procesora koji se može iskoristiti za obavljanje nekog korisnog posla. Također je bitna razlika da DMA radi blokovski prijenos podataka te je neovisna o procesoru, što nije slučaj dok koristimo prekidni mehanizam. DMA će tražiti od procesora prekid po bloku podataka, dok će prekidni mehanizam za svaki poslani znak tražiti zaseban prekid.

4. POGLAVLJE

29. Što je zajedničko procesu roditelju i procesu djetetu?
Ništa. Dijete dobiva kopije instrukcija i podataka... svaki proces dobiva svoj adr.prostor(osim, kako ti je opisano, u slučajevima kad se instrukcijski dio dijeli) i ne mogu jedno drugom adresirat varijable.

30. Objasniti pojmove višezadaćni rad, višedretveni rad i višekorisnički rad! (OK)
Višezadaćni rad (multitasking) – mogućnost računala da obavlja istovremeno više poslova
Višedretveni rad (multithreading) – brzo obavljanje više procesa u jednom programu
Višekorisnički rad (multiuser system) – računalo koje mogu koristiti više osoba
svima je zajedničko da obavljaju više zadaća odjenom

31. Koja sredstva dijele dretve istog procesa? (OK), 04-05
Dretve koje djeluju unutar jednog procesa dijele sva sredstva koja je operacijski sustav stavio na raspolaganje tom procesu. Posebice se to odnosi na adresni prostor procesa.

32. Kako je podijeljen spremnički prostor procesa/dretve? (OK), 04-10
Dretveni prostor: dio za instrukcije dretve, dio za stog dretve i dio za lokalne podatke dretve.

Spremnički prostor: se sastoji od više dretvenih prostora i zajedničkog prostora (koji mogu dohvaćati sve dretve procesa – globalne varijable).

33. Kako najjednostavnije dretve mogu komunicirati?

- tako da jedna dretva piše direktno u domenu druge
- preko rezerviranog dijela zajedničkog spremnika (zajedničke varijable)

34. Navesti i pojasniti uvjet nezavisnosti dretvi? (OK), 04-15
Dretve i su nezavisne kad vrijedi:

$$\cap \quad \cup \quad \cap \quad \cup \quad \cap \quad = \emptyset$$

Dvije dretve su nezavise kada se mogu obavljati u isto vrijeme bez da jedna drugu ometa – to znači da se ne smije dogoditi da jedna dretva piše na adrese koje bi druga dretva mogla koristiti (za čitanje ili pisanje). Dakle, smiju samo čitati s istih adresa da bi bile nezavisne.
Odnosno: ako je bilo koji od ova tri uvjeta ispunjen, dretve su zavisne.

- $\cap$ - u isto vrijeme i-ta dretva čita podatak i j-ta dretva zapisuje – ako se to dogodi – zavisne
- $\cap$ - u isto vrijeme j-ta dretva čita podatak i i-ta dretva zapisuje – ako se to dogodi – zavisne
- $\cap$ - u isto vrijeme i i-ta i j-ta dretva zapisuju podatke na istu kodomen – zavisne

35. Kako se u usmjerenom grafu sustava dretvi prepoznaju zavisne, tj. nezavisne dretve? (OK)

Dretve koje se nalaze na istom putu od početne do završne dretve moraju se izvoditi propisanim redoslijedom, te su dretve međusobno zavisne.
Dretve koje se ne nalaze na istom putu moći će se izvoditi proizvoljnim redoslijedom i one moraju biti međusobno nezavisne.

36. Navedite uvjete koje moraju zadovoljavati algoritmi za međusobno isključivanje? (OK)

- 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.

37. S koliko dretvi radi Dekkerov, a s koliko Lamportov algoritam? (OK)
Dekkerov algoritam radi sa dvije dretve, dok Lamportov algoritam služi za isključivanje N dretvi.

38. Koje strukture podataka koriste Dekkerov i Lamportov algoritam? (OK)
Dekker - najobičnije varijable (dvije zastavice ZASTAVICA[I] i ZASTAVICA [J] i pravo PRAVO)
Lamport - polje zastavica (koje govori koja dretva pokušava ući) ULAZ[I], polje brojeva dretvi BROJ[J] i varijabla zadnjeg broja ZADNJI_BROJ.

39. Koji je najjednostavniji način međusobnog isključivanja na 1-procesorskom računalu? (OK)
Prekidom.

40. Kako je osmišljen višedretveni rad u jednoprocesorskom računalu? (OK)

Višedretveni rad se može provesti tako da procesor naizmjence izvodi sve dretve. Pri prekidanju izvođenja jedne dretve njen sadržaj se mora pohraniti izvan procesora i u registre staviti sadržaj dretve koja se želi pokrenuti. Time nismo postigli brzinu, ali smo omogućili da se može izvoditi jedna dretva dok druga iz nekog razloga mora čekati.

41. Kako je moguć višeprocesorski rad na jednoprocesorskom računalu?
Višeprocesorski rad zahtjeva više procesora. Višedretveni rad je moguć na jednom procesu tako da se naizmjenično izvršava malo jedna dretva, malo druga, malo treća itd.

42. Navedite instrukcije procesora koje služe kao sklopovska potpora međusobnom isključivanju? (OK)
TAS (test & set) - funkcija koja se obrađuje u dva nedjeljiva sabirnička ciklusa pa je osigurano da s takvim pozivom neće dvije dretve obaviti istu stvar (dohvatiti isti podatak i promijeniti ga!)
SWAP (zamijeni) - 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 – dohvati i povećaj za 1 - u prvom ciklusu dobave sadržaj adresirane lokacije, i smjeste ga u jedan od registara procesora, a u drugom ciklusu pohranjuju u tu lokaciju taj sadržaj uvećan za jedan.

43. Ostvariti međusobno isključivanje pomoću XY instrukcije (iz 40. pitanja)! (OK)

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

5. POGLAVLJE

44. Što predstavlja pojam ulazak u jezgru OS-a i kad se događa? (OK)
Pojam predstavlja poziv jezgrine funkcije i zbiva se prekidom.

45. Što se događa prilikom poziva jezgrine funkcije? (OK)
Onemogući se daljnje prekidanje (ne mogu se pozivati druge funkcije). Pohranjuje se kontekst, pamti se gdje je bio prekid. Poziva se odgovarajuća funkcija koja sa sustavnog stoga pohranjuje konktekst u opisnik aktivna dretva.

46. Što se radi prilikom izlaska iz jezgre? (OK)
Aktivira se prva dretva iz reda Pripravne D. u red Aktivna D. Obnovi se kontekst iz opisnika Aktivna D. Omogućuje se prekidanje, vraća se iz prekidnog načina. Obnavlja se PC. Iz sustavskog se prelazi u korisnički način rada.

47. Opisati način ostvarenja i osnovnu upotrebu binarnog semafora. (OK)
Binarni semafor je jezgrin mehanizam koji služi za međusobno isključivanje dretvi, tj. zaštitu kritičnih odsječaka. On propušta jednu po jednu dretvu.

48. Od čega se sastoji jezgra OS-a? (OK)
Od struktura podataka i jezgrinih funkcija.

49. Navesti vrste prekida. (OK)
- sklopovski prekidi
- programski prekidi
- prekidi od sata.

50. Navesti podatke dretve (sadržaj opisnika dretve). (OK) 05-13
- kazaljke 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 spremnika prostora
- veličina dretvenog spremničkog prostora
- adresa prve instrukcije
- zadano kašnjenje
- prostor za smještanje konteksta

51. Navesti strukture podataka jezgre. (OK)
1. **Lista (red) postojećih dretvi** – kada se dretva nalazi na samo jednoj listi, onda je u pasivnom stanju
2. **Lista (red) aktivnih dretvi** – dretve koje se izvode, broj članova u toj listi jednak je broju procesora. U jednoprocesorskom sustavu ima samo jedan deskriptor.
3. **Lista (red) pripravnih dretvi** – ako se ne izvode, a spremne su. Prema načinu formiranja red može biti – po redu prispjeća (FCFS (first come, first served) ili FIFO) ili prioritetni
4. **Red semafora** – dretve tijekom izvođenja mogu biti blokirane čekajući na ispunjenje nekog uvijeta (4 načina blokiranja čekanjem na: BSEM (binarni semafor), OSEM (opći semafor), istek zadanog intervala kašnjenja, završetak U/I operacije
5. **Lista (red) odgođenih dretvi** – zadano kašnjenje dretve
6. **Lista (red) U/I naprave** – redova koliko ima U/I naprava, U/I naprave se obično koriste pojedinačno

latentna dretva,
opisnik za svaku dretvu
Napomena – ovdje sam očito nešto dodavao, čim nije lijepo formatirano znači da sam naknadno nešto editirao (latentna dretva i opisnik za svaku dretvu). Pa provjerite još ovo zadnje, na kaj sam mislio. :D

52. Koja su blokirana stanja dretvi? (OK)

- čekanje na binarnom seamforu
- čekanje na općem semaforu
- čekanje na istek zadanog intervala kašnjenja (odgođene dretve)
- čekanje na završetak U/I operacije.

53. Koja su moguća stanja dretvi? (OK, Pogledati i graf 05-35!!!)

- aktivna dretva
- pasivna dretva
- BSEM (binarni semafor)
- OSEM (opći semafor)
- odgođene dretve
- čekadnje na završetak U/I operacije
- pripravne dretve

54. Koje vrste prekida uzrokuju jezgrine funkcije Započeti U/I i Prekid U/I? (OK)

Sklopovski prekid.

55. Može li se kontekst dretve koja obavlja neku jezgrenu funkciju naći u opisniku u listi aktivna dretva, i zašto? (OK)

Ne. Lista aktivnih dretvi sadrži dretvu koja se treba početi izvršavati nakon dretve jezgrene funkcije.
Ne, jezgrina funkcija ne može biti prekinuta.

56. Što obavlja instrukcija Aktivirati dretvu iz reda pripravne dretve? (OK)

Premješta opisnik dretve iz reda Pripravne_D u red Aktivna_D i nakon toga se iz tog opisnika prenosi kontekst dretve u registre procesora.

57. Čemu služe jezgrini mehanizmi binarni i opći semafor? (OK)

Služe za međusobno isključivanje dretvi. Jedan od drugoga se razlikuju samo u realizaciji.

58. Što je najveći nedostatak međusobnog isključivanja? (OK)

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

59. Koje strukture podataka koriste BSEM, OS, OSEM? (OK)

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

60. U pseudokodu napisati jezgrine funkcije Čekaj_BSEM (ili OS, ili OSEM) i Postavi_BSEM(ili OS, ili OSEM)?

SKRIPTA + ono sa FER.HR

61. Kako treba proširiti jegrine funkcije u višeprocesorskom sustavu? (OK), vidi 05-80

Jezgrine funkcije treba proširiti radnim čekanjem.

Činjenice
Komunikacija među dretvama je znatno brza od komunikacije među procesima.
Komunikacija među dretvama istog procesa odvija se bez uplitanja OS
Jedinstevno za svaku dretvu: Thread ID (TID), stanje registra, kazaljka stoga (stack pointer), stog, prioritet
Skup zauzetih sredstava je isti za sve dretve jednog procesa.
U programu s više procesa, globalne varijable u programu vide svi procesi ali prilikom njihove promjene i jedan i drugi proces dobivaju kopije tih varijabli.

Napomene vezane za prvi međuispit:
U pitanjima "što procesor radi kada dođe prekid / poziv potprograma / što procesor konstanto radi / itd." valja odgovarati pseudokodom.
Što se tiče onih 5 neuspjelih primjera za ostvarivanje međusobnog isključivanja, ne treba znati kod, ali treba znati zašto ne valjaju!

2. CIKUS PREDAVANJA

POJMOVI

Pravila za pisanje korisničkih funkcija:

1) monitorska funkcija je zaštićena monitorskim semaforom (uđi u monitor, izađi iz monitora)
2) uvjet se ne ispituje naredbom ako je već dok je
3) dretva koja oslobađa neku drugu dretvu iz reda uvjeta, nakon što ju je oslobodila završava
4) jezgrine funkcije za ostvarenje monitora koriste se isključivo unutar monitorskih funkcija
5) nakon uvrsti_u_red_uvjeta obavezno napisati uđi_u_monitor

Problemi sa semaforima

1) semafor ispituje jedan uvjet, odnosno služi međusobnom isključivanju, sinkornizaciji i brojanju događaja
2) Ispitivanje semafora povezano je sa zauzećem sredstva kojeg semafor štiti – ne može se razdvojiti ispitavnje od semafora.
3) opasnost od potpunog zastoja

Funkcije za monitore

M-varijabla zaključvanja
mutex_lock (M) – uđi u monitor
mutex_unlock (M) – izađi iz monitora
cond_wait (red uvjeta, M) – uvrsti u red uvjeta, uđi u monitor
cond_signal (red_uvjeta) – oslobodi iz reda uvjeta, uđi u monitor
cond_broadcast
POGLEDATI ONU NJIHOVU SKRIPTU SA FERWEB-A ZA VIŠE!

pojmovi:

- iskoristivost (opterećenje) – ρ
- broj poslova koji ulaze u sustav u jedinici vremena – α
- broj poslova koje poslužitelj može obaviti u jedinici vremena – β
- prosječan broj poslova u sustavu - n
- prosječno zadržavanje posla u sustavu (kvaliteta usluge) - ρ
- Littleovo pravilo - $n = \alpha \cdot T$ (riječima!)
- Markovljev lanac (skicirati)

U kružnom posluživanju dretvama višeg prioriteta može se dodjeljivati dva ili više uzastopna kvanta procesorskog vremena

PITANJA

1. Kako se pozivaju jezgrine funkcije? OK (05-06)

Poziv jezgrine funkcije — ulazak u jezgru — zbiva se pod utjecajem sklopovskog ili programskog prekida

U jednoprocesorskom sustavu je time osigurano međusobno isključeno obavljanje jezgrinih procedura

2. Od čega se sastoji jezgra? OK (05-06)

Jezgra se sastoji od jezgrinih struktura podataka i jezgrinih funkcija (pozivaju se kod prekida).

3. Što sadrži opisnik dretve? (OK) 05-13

- kazaljke 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 spremnika prostora
- veličina dretvenog spremničkog prostora
- adresa prve instrukcije
- zadano kašnjenje
- prostor za smještanje konteksta

4. Navedi strukture podataka jezgre. (OK)

Liste (redovi) dretvi i opisnici dretvi.

7. Lista (red) postojećih dretvi – kada se dretva nalazi na samo jednoj listi, onda je u pasivnom stanju
8. Lista (red) aktivnih dretvi – dretve koje se izvode, broj članova u toj listi jednak je broju procesora. U jednoprocesorskom sustavu ima samo jedan deskriptor.
9. Lista (red) pripravnih dretvi – ako se ne izvode, a spremne su. Prema načinu formiranja red može biti – po redu prispjeća (FCFS (first come, first served) ili FIFO) ili prioritetni
10. Red semafora – dretve tijekom izvođenja mogu biti blokirane čekajući na ispunjenje nekog uvijeta (4 načina blokiranja čekanjem na: BSEM (binarni semafor), OSEM (opći semafor), istek zadanog intervala kašnjenja, završetak U/I operacije
11. Lista (red) odgođenih dretvi – zadano kašnjenje dretve
12. Lista (red) U/I naprave – redova koliko ima U/I naprava, U/I naprave se obično koriste pojedinačno

5. Koja su blokirna stanja dretve? (OK) 05-26

- čekanje na binarnom seamforu
- čekanje na općem semaforu
- čekanje na istek zadanog intervala kašnjenja (odgođene dretve)
- čekanje na završetak U/I operacije.

6. Opiši kako radi binarni semafor. (OK) 05-27

- omogućuje nam ostvarenje međusobnog isključivanja dretvi, a sastoji se od varijable Bsem[I].v koja igra ulogu zastavice i kazaljke koja pokazuje na red opisnika dretvi koje nisu uspjele proći semafor
- semafor može poprimiti dvije vrijednosti 0 i 1. Ako je Bsem[I].v == 1, semafor je prolazan, ako je Bsem[I].v == 0 semafor je neprolazan.
- dretva koja naiđe na prolazni semafor bit će propuštena i jezgra će staviti vrijednost 0 u Bsem[I].v
- ako jedna ili više dretvi pokušaju proću uz neprolazni semafor, one će biti zaustavljene i njihovi će opisnici biti prebačeni u red pridružen tom semaforu.

7. Opiši kako radi opći semafor. 05-29

8. Skiciraj graf stanja dretva SLAJDOVI 05-35,

9. Što obavlja instrukcija "aktiviraj prvu dretvu"? (OK)
premjestiti prvi opisnik iz reda Pripravne_D u red Aktivna_D;
obnoviti kontekst iz opisnika Aktivna_D;
omogućiti prekidanje;
vratiti se iz prekidnog načina;

10. Čemu služe semafori?
Služe za međusobno isključivanje dretvi

11. Koju strukturu podataka ima binarni i opći semafor?
BSEM koristi varijablu Bsem[i].v i kazaljku
OS koristi varijablu OS.v i kazaljku
OSEM koristi varijablu Osem[J].v i kazaljku.

12. Na koji način se obavljaju jezgrine funkcije na jednojezgrenom procesoru, a kako na višejezgrenom procesoru?
U 1-CPU se obavljaju međusobno isključivo (pomoću zabranjivanja prekida)
U višejezgrenom – TAS?
Nije mi ovo najjasnije pitanje.

13. Naizmjenično obavljanje 2 dretve
Se može sinkronizirati upotrebom minimalno 2 binarna semafora (mogu i OS i OSem)

14. Navedi načine međusobnog isključivanja (programske)?
Dekkerov postupak – Međusobno isključivanje 2 dretve
Petersonov postupak – Međusobno isključivanje 2 dretve (pojednostavljen Dekker)
Lamportov protokol – Međusobno isključivanje većeg broja dretvi

15. Što je to potpuni zastoj?
Blokiranje reda u redu čekanja , stanje sustava u kojem su sve dretve blokirane na nekom redu uvjeta

16. Koji su nužni uvjeti za potpuni zastoj? (OK) 05-29
Potpuni zastoj se može dogoditi kada se najmanje dvije dretve nadmeću za najmanje dva sredstva. Tri su nužna uvjeta za nastajanje potpunog zastoja:

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

17. Što je to monitor?
Monitor je sinkronizacijski mehanizam u višedretvenim/višeprocesorskim sustavima. U njemu su objedinjeni mehanizmi suradnje dretvi kao prikladne nakupine funkcija koje djeluju na njegovu strukturu podataka.
18. Navesti jezgrine strukture podataka potrebne za ostvarenje monitora!
Red čekanja Monitor[M], red čekanja Reduvjeta[M,K] gdje svaki red predstavlja jedan uvjet
Funkcije su: Uđi_u_monitor, Izađi_iz_monitora, Uvrsti_u_red_uvjeta, Oslobodi_iz_reda_uvjeta

19. Kada se mogu naći 2 dretve u monitoru?
Kada dretva koja je ušla u monitor može djelovat na ispunjenje nekog uvjeta tako da time deblokira dretvu koja je unutar monitora čekala na ispunjene tog uvjeta.
Napomena: Oko ovog pitanja sam čuo dosta svađe jer neki tvrde da je odgovor ovo gore, neki tvrde da je to samo prividno (da se dvije nalaze u monitoru), a neki kažu da jednostavno NIKAKO. Pa vi budite pametni…

20. Kojim jezgrinim mehanizmima je zaštićen monitor?
Čekanjem monitorskog semafora

21. Što je to monitorska funkcija?
- monitorska funkcija se odvija u korisničkom načinu rada (kreira je korisnik)
- unutar monitorske funkcije dretva ispituje složeni uvijet

22. U čemu se razlikuje monitorski semafor od binarnog semafora?
Monitorski semafor omogućuje i drugim monitorskim funkcijama da koriste istu strukturu podataka.
- kod BSEM imamo samo dvije funkcije (čekaj/postavi)
- MON 2 primarne funkcije + dodatne (2) (uđi/izađi) + (uvrsti u red uvjeta / oslobodi iz reda uvjeta)

23. objasniti parametre sustava:

a) iskoristivost/faktor iskorištenja: ro=alfa/beta, a u postocima: ni=(Tp/Td)*100 (%) ili alfa/beta*100%
b) alfa - broj poslova koji ulazi u sustav u jedinici vremena
c) beta - broj poslova koje poslužitelj može obaviti u jedinici vremena
d) n potez - prosjećan broj poslova u sustavu
e) T potez - prosjećno zadržavanje u sustavu

24. Navesti jezgre funkcije za ostvarenje monitora!
Uđi_u_monitor, Izađi_iz_monitora, uvrsti_u_red_uvjeta, oslobodi_iz_reda_uvijeta

25. Objasni:
Littleovo pravilo
Markoljev lanac(skiciraj i objasni)

26. Skicirajte Markovljev lanac gdje stanja predstavljaju broj poslova u sustavu. Sustav ima Poissonovu razdiobu. To si nađite u slajdovima

27. U kojem slučaju se može dozvoliti veliki faktor iskorištenja?
Kada su trenutci dolaska i trajanja poslova deterministički (daju se odrediti i stalno su isti)
U Determinističkom, - ako trenuci dolazaka poslova i trajanje obrade odgovaraju determinističkoj raspodjeli, inače dolazi do zagušenja.

28. O čemu ovisi parametar β?
O vrsti posla, o poslužitelju

29. Koja je zadnja aktivnost unutar monitorske funkcije?
Zadnja aktivnost unutar monitorske funkcije je izaći iz monitora ili oslkoboditi dretvu iz reda uvjeta.

3. CIKLUS PREDAVANJA

8. POGLAVLJE

1. Gdje se generiraju adrese unutar procesora? OK

U von Neumannovoj arhitekturi računala se sve adrese generiraju unutar procesora, i onda se one neposredno preko adresnih vodiča dovode do spremnika.

a) **adrese instrukcija** koje dolaze iz **programskog brojila** kad se iz radnog spremnika dohvaćaju instrukcije

b) **stogovne adrese** koje dolaze **iz registra kazaljke stoga** kad se adresira stog;

c) **adrese operanada** i rezultata operacija (adrese podataka), koje se stvaraju na temelju **sadržaja adresnih dijelova instrukcija** kad se adresiraju podaci.

2. Na koje segmente se dijeli procesni adresni prostor? (Sumnjivo, ali nisam našao bolji)

Svaki računalni proces ima barem jednu dretvu. Ako se proces sastoji od više dretvi, onda se spremnički prostor procesa dijeli na dretvene prostore i jedan zajednički prostor u koji mogu pristupiti sve dretve. Unutar svakog dretvenog podprostora može se prepoznati:

a) **kodni** ili **instrukcijski** segment u koji se pohranjuju instrukcije programa ili programski kod, i adresira se iz programskog brojila;

b) **stogovni segment** u koji se pohranjuje stog, i adresira se iz registra kazaljke stoga;

c) **podatkovni segment** u koji se pohranjuju podaci, i cije se adrese stvaraju na temelju sadrzaja adresnih dijelova instrukcija.

3. Koliko iznosi ukupno trajanje prijenosa podataka između tvrdog diska i RAM-a? (THX Daxxx)

I.)trajanje postavljanja glave:

a)trajanje traženja (seek time) – t ubrzavanja ručice glave, gibanje konstantnom brzinom, t usporavanja, t finog pozicioniranja

b)rotacijsko kašnjenje (avg Tr/2)

II.)trajanje prijenosa podataka – ovisi o brzini prijenosa i količini podataka

Vp=broj_sektora*veličina_sektora/Tr

4. Čime je određeno trajanje prijenosa podataka? FER2

brzinom prijenosa (engl. data transfer rate)

Brzina prijenosa određena je brzinom kojom ispod glave za čitanje ili pisanje promiču bajtovi sektora. Može se izračunati tako sa se broj sektora na stazi pomnoži s brojem bajtova u sektoru i taj umnožak podijeli s trajanjem jednog okretaja.

količinom prenešenih podataka

mislim da ovo ne treba dodatno pojasnjavati :)

ponašanjem upravljačkog sklopa diska.

Spremnik upravljačkog sklopa služi kao međuspremnik između radnog spremnika i samog magnetskog diska. Upravljački sklop obavlja sve detalje komuniciranja s diskom. U njemu se može obaviti i zaštitno kodiranje odnosno dekodiranje i eventualno ponovno čitanje s diska kako bi se eliminirale moguće pogreške nastale od trenutnih smetnji. Brzina prijenosa iz upravljačkog sklopa diska u radni spremnik ovisi i o brzini prijenosa na sabirnici, kao i o trenutnom intenzitetu prometa na sabirnici.

U najgorem slučaju upravljački sklop može djelovati tako da ukupnom trajanju prijenosa pridonosi dijelove milisekunde i mi ćemo ga, u pojednostavljenom gledanju, naprosto zanemariti.

5. Od čega se sastoji trajanje postavljanja glave?
Trajanje traženja staze (4 komponente: trajanje ubrzavanja ručice glave, kretanje konstantnom brzinom, trajanje usporavanja i trajanje finog pozicioniranja)
Rotacijsko kašnjenje (čeka se da se nakon postavljanja glave na odabranu stazu ispod glave pojavi traženi sektor. Max=>ako je traženi sektor taman prolazio ispod glave kad je ona prispjela u stazu; mora se čekati puni okretaj. Min=>kad se traženi sektor pojavi u trenutku njezina postavljanja na stazu)

6. Od čega se sastoji trajanje traženja staze? OK
Trajanje traženja staze (seek time, trajanje traženja) se sastoji od:
a) trajanje ubrzavanja ručice glave
b) kretanje konstantnom brzinom
c) trajanje usporavanja
d) trajanje finog podešavanja (pozicioniranaja)

7. Uobičajeno se za vrijeme traženja uzima vrijeme koje je potrebno za prijelaz preko ____trećine_____ ukupnog broja staza. OK

8. Zbog čega nastaje rotacijsko kašnjenje i o čemu ovisi? OK
Rotacijsko kašnjenje 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.
Najlošiji slučaj – adresirani sektor je upravo prolazio ispod glave kad je ona prispjela na stazu
Najbolji slučaj – adresirani sektor upravo u trenutku postavljanja glave nailazi ispod glave.
Prosječno – uzimamo vrijednost tih dvaju krajnjih slučajeva i pretpostavljamo da je prosječno rotacijsko kašnjenje jednako poloviti trajanja jednog okretaja diska – Tr/2.

9. Navedite sadržaj procesnog informacijskog bloka! (THX Daxxx)
Tablice sektora, podaci o smještaju programa u radnom spremniku, opisnici dretvi procesa, opisnik virtualnog procesnog adresnog prostora.

10. Objasniti statičko i dinamičko raspoređivanje spremnika! OK
Statičko - 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. Nedostatak – unutarnja i vanjska fragmentacija
Dinamičko - program izračunava adrese kao da kreće od 0 (logički adresni prostor), smjestiti će u radni spremnik koji započinje fizičkom adresom PA, u procesoru iza adresnog međuregistra je jedno sklopovsko zbrajalo i uz njega registar u koji se zapisuje početna adresa fizičkog adresnog prostora, 'reg' se naziva bazni registar. Programi se slažu jedan iza drugoga. Registar ograde - koristi se da program ne bi zauzeo prostor susjednog programa. Računa se kao OG=PA (početna adresa u fizičkom adresnom prostoru) + VELIČINA (veličina programa). Ovo rješenje je sklopovsko i registar ograde se provjerava prije ulaska u radni spremnik. I dalje nedostatak fragmentacija (vanjska) - pojava rupa (kada se neki program makne iz radnog spremnika).

11. Navedite vrste fragmentacije (statičko i dinamičko)! OK
Prilikom statičkog raspoređivanja radnog spremnika javljaju se dvije vrste fragmentacija:
Unutarnja (internal fragmentation) – programi neće biti jednake veličine kao particije te će dijelovi particija ostati neiskorišteni
Vanjska (external fragmentation) – tijekom rada se može dogoditi da svi procesi čiji su programi smješteni u istu particiju bivaju blokirani pa ta particija radnog spremnika ostane 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.

Kod dinamičkog raspoređivanja radnog spremnika imamo samo vanjsku fragmentaciju:
Problem fragmentacije spremničkog prostora – tijekom vremena neki procesi završavaju izvođenje i oslobađaju svoj spremnički prostor. Drugi procesi bivaju blokirani i izbacuju se iz radnog spremnika pa tako također oslobađaju svoj procesni prostor kako bi načinili mjesta za programe procesa koji se mogu izvoditi. Ti će se procesi kasnije vratiti u spremnik ali najvjerojatnije na neko drugo mjesto. S obzirom da veličine programa nisu jednake u spremniku će se nakon nekog vremena između procesnih adresnih prostora pojaviti prazni dijelovi spremnika koje nazivamo rupama.

12. Kako se može ublažiti problem fragmentacije prilikom dinamičkog dodjeljivanja spremnika, s obzirom da se ne može se izbjeći? OK
Pri svakom oslobađanju nekog procesnog prostora novonastala rupa spaja 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
Ako fragmentacija postane prevelika postoji mogućnost da se privremeno obustavi izvođenje dretvi i presloži programe u kompaktni prostor. Ta se tehnika zove „garbage collection“

13. Navesti i dokazati Knuthovo 50%-tno pravilo! OK, izvod – Daxxx fotka, FER2, skripta
U stacionarnom stanju u spremniku postoji broj rupa koji je jednak polovini broja punih blokova.

14. Kad se ustanovi da je fragmentacija prilikom dinamičkog dodjeljivanja spremnika prevelika, tada se _zaustave se sve dretve, presloži se sve programe u kompaktni prostor, i krene se dalje (garbage collection)___ **. OK**

15. Kojim redoslijedom mogu biti smještene stranice logičkog adresnog prostora u okvire fizičkog spremnika? ??? Ovo sam izvadio sa materijala i nisam našao u skripti
Proizvoljnim redoslijedom.

16. Objasniti preklopni način uporabe spremnika! OK, FER2
Osnovna zamisao je da se program čiji je logički adresni prostor veći od fizičkog spremnika podjeli na dijelove koji ne moraju istovremeno biti spremljeni u radnu memoriju (fizički spremnik). U radnom spremniku mora se nalaziti onaj dio programa koji se trenutno izvodi kao i dio podatkovnog prostora koji te instrukcije adresiraju. Program, treba podijeliti na jedan osnovni dio koji se uvijek nalazi u radnom spremniku i na dijelove koji se naizmjence smještaju u preostali dio korisničkog dijela fizičkog spremnika. Zamjenljivi dijelovi programa se u preklopu (engl. overlay) smještaju u radni spremnik pa taj način nazivamo preklopnim načinom uporabe spremnika.
Programer mora podijeliti program na odgovarajuće dijelove i u osnovni dio programa ugraditi mehanizme za donošenje odluke o pravodobnom prebacivanju preklopnih dijelova u radni spremnik.
Postupak podjele programa na dijelove i način njihova smještanja u radni spremnik treba automatizirati, kako bi se olakšala priprema programa i izbjegle pogreške programera.
U današnjim računalnim sustavima upotrebljavaju se procesori s prikladnim spremničkim međusklopovima koji omogućuju da operacijski sustavi ostvare vrlo djelotvorno gospodarenje spremničkim prostorom.

17. Kako treba podijeliti program u preklopnom načinu uporabe radnog spremnika? OK
U radnom spremniku mora se nalaziti onaj dio programa koji se trenutno izvodi kao i dio podatkovnog prostora koji te instrukcije adresiraju. Pritom program, treba podijeliti na jedan osnovni dio koji se uvijek nalazi u radnom spremniku i na dijelove koji se naizmjence smještaju u preostali dio korisničkog dijela fizičkog spremnika.

18. Kako se dijeli logički adresni prostor, a kako fizički? OK
Logički adresni prostor dijelimo na **stranice**, a **fizički** adresni prostor dijelimo na **okvire**.

19. O čemu ovisi veličina logičkog adresnog prostora, a o čemu veličina fizičkog adresnog prostora? OK
Veličina logičkog adresnog prostora ovisi **o arhitekturi procesora** (32 bitna, 64 bitna…), a **veličina fizičkog adresnog prostora** o **količini radne memorije** (RAM-a).

20. Čemu sluzi tablica prevođenja, i od kojih se elemenata sastoji? (THX Daxxx)
Tablica prevođenja je sastavni dio spremničkog međusklopa u kojoj se vodi evidencijao tome u kojem je okviru smještena pojedina stranica (za više vidi TLB kod Intel x86). Kaže daxxx - od redaka (za svaku stranicu jedan redak) u kojima se pohranjuje broj okvira u kojem se nalazi (ako je in) te dodatnog bita prisutnosti za svaki redak]

21. U čemu se razlikuju prekidi izazvani zbog promašaja stranice od ostalih vrsta prekida? OK
Prekid uslijed promašaja stanice dogodio se za vrijeme izvođenja instrukcije, a ne na kraju instrukcije. Stoga se instrukcija koja je izazvala prekid mora ponoviti. Da bismo ponovili instrukciju, moramo znati sadržaje svih registara prije izvođenja te instrukcije. Zbog toga procesori predviđeni za ostvarivanje virtualnog spremnika imaju posebne skrivene kopije registara u kojima se čuvaju nepromijenjene vrijednosti registara tijekom izvođenja cijele instrukcije. Vrijednosti tih registara se obnavljaju tek na kraju izvođenja svake instrukcije.

22. Čemu služi posmačni registar povijesti? OK
Dio praktične aproksimacije LRU strategije zamjene stranica. U tablicu stranica dodani su posmačni registri u koje možemo posmaknuti bitove pristupa. Dio operacijskog sustava za gospodarenje spremničkim prostorom djeluje na sljedeći način: Pri punjenju stranice u neki okvir briše se njezin bit pristupa i pripadni posmačni registar. Prekid od sata (periodno) posmakne sve bitove pristupa u registre i zatim ih izbriše. Ponovni pristup (čitanje ili pisanje) do stranice postavit će opet njezin bit pristupa u 1. Nakon nekoliko perioda otkucaja sata, sadržaj posmačnih registara može se pročitati kao binarni broj. Manji broj pokazuje da se u stranicu dulje vremena nije pristupalo. Kada se pojavi potreba za izbacivanjem stranice izbacuje se ona sa najmanjim brojem (ili jedna iz razreda sa jednakim najmanjim brojevima).

23. Opisati satni algoritam. OK
Satni algoritam je inačica LRU strategije. Upotrebljava se u UNIX i Windows NT serijama operacijskih sustava. Postupak djeluje na sljedeći način – sve se stranice u listu svrstavaju po redu prispijeća. Lista se obilazi kružno posebnom kazaljkom (kada dođe do kraja kazaljka se vraća na početak liste – otuda i naziv postupka). Kada se pojavi potreba za praznim okvirom izbacit će se stranica na koju pokazuje kazaljka ako je njezina zastavica A (zastavica pristupa) jednaka nuli. Stranica će se preskočiti ako je vrijednost zastavice A jedan, ali će se pritom njezina vrijednost promijeniti u 0. Kazaljka se pomiče za po jedno mjesto dok ne naiđe na stranicu sa A jednakoj nuli. Ako kazaljka kružno obiđe cijelu listu pronalazeći sve same jedinice ona će ih sve obrisati te izbaciti stranicu s koje je započela obilazak.

28. Kakva je sklopovska potpora za straničenje u Intel x86 arhitekturi? Također koliko pristupa radnom spremniku je potrebno ako je stranica navedena u TLB i ako stranica nije navedena u TLB. (THX Daxxx) i moja i Daxxxova preporuka – učite to iz skripte!

Unutar procesora – CR3 registar – kazaljka na tablicu prevođenja (svaki proces ima svoju tablicu)
Tablica prevođenja – dvorazinska

a) direktorij stranica – 1024 kazaljke po 32 bita pokazuju na stranice pod b)
b) tablice stranica – 1024 riječi po 32 bita

AL (logička adresa):

- gornjih 10b izabire iz a) kazaljku na jednu od tablica stranica
- sljedećih 10b iz te tablice odabire adresu stranice
- donjih 12b ide izravno u donjih 12 AF (fizičke adrese.)

Tako odabranih 32b → 1 od registara TLB-a (priručni međuspremnik za prevođenje adresa)
TLB – gornjih 20b ide u gornjih 20b AF (sadrži i dodatni 20b registru koji ide gornjih 20b AL, za usporedbu s novom adresom - ubrzanje.)

9. POGLAVLJE

24. Što sačinjava opisnik datoteke? OK
Naziv datoteke, tip datoteke, lozinka, ime vlasnika, pravo pristupa, vrijeme stvaranja, vrijeme zadnje uporabe, ime posljednjeg korisnika, opis smještaja datoteke

25. Što sačinjava datotečnu tablicu? OK
Veličina diska (particije), količina slobodnog prostora, informacije o slobodnom prostoru (pozicija i sl) opisnici datoteke.

2. Na koji način je opis smještaja pohranjen u NTFS, a na koji u UNIX sustavu? (THX Daxxx)

NTFS
diskovni prostor dodjeljuje se po skupinama sektora (1,2,4,8)/cluster;
referenca na disk uporabom LCN (r.br.clustera);
postoji datoteka MFT koja sadrži opisnik svake datoteke (1 zapis je veličine 1 clustera ~4KB);
male datoteke se cijele smještaju unutar MFT zapisa;
veće datoteke – zapis sadrži indekse clustera (+ dodatna potrebna proširenja);
datoteka se dijeli na „virtualne skupine“ („stranice“ jednake veličini „okvira“ (clustera))
– r.br.str. = VCN;

UNIX
13 kazaljki:
10 neposrednih - pokazuju na prvih 10 sektora [dat. do 10KB]
1 jednostruko indirektna - pokazuje na sektor s idućih 256 kazaljki [do 266KB]
1 dvostruko indirektna - pokazuje na sektor s kazaljkama na 256 sektora od kojih svaki ima 256 kazaljki () [do 64MB]
1 trostruko indirektna - kao gore samo još jedan stupanj indirekcije (256x256x256) [do 16GB]

10. POGLAVLJE

26. Po kojim pravilima djeluje globalni logički sat? OK

- proces Pi povećava svoj logički sat Ci između svaka svoja dva događaja
- kada proces Pi šalje poruku m on u nju pridodaje vremensku oznaku Tm=Ci
- kada proces Pj primi poruku on postavlja u svoj logički sat Cj vrijednost koja je veća od njegove prethodne vrijednosti i veća od prispjele vremenske oznake Tm.

27. Zašto izgladnjivanje nije moguće kod Ricarta&Agrawala te kod Lamporta? FER2 – provjeriti

Zbog globalnog logičkog sata.

PROTOKOLI

- slajdovi pred sam kraj (konkretno počinje na 10-51) pa to lijepo proučite.

Besramno pokradeno od Daxxxa:

1. Centralizirani protokol

Jedna od čvorova odgovoran za ostvarenje međusobnog isključivanja, u njemu su svi podaci, on daje dozvole za ulazak u KO
Pi čija dretva želi u KO šalje zahtjev(i) centralnom čvoru (pa čeka odgovor);
c.č. prihvaća zahtjev i šalje odgovor(i) čvoru Pi kad ovaj smije ući u KO;
nakon KO dretva šalje izlaz(i) c.č.-u;
(dozvola ulaska ~ značka)

2. Protokol sa putujućom značkom

Jedna značka kao poruka putuje kroz čvorove;
kad prispije u čvor Pi, on:

a) zadrži je ako želi u KO i šalje dalje po završetku
b) šalje dalje ako ne želi u KO

3. Lamportov protokol

(nabrojat ću pravila)

a) Pi zahtjeva ulazak u KO – stavlja poruku zahtjev(i,T(i)) u svoj red, šalje je svim procesima
b) kad Pj primi zahtjev(i,T(i)) – uskladi svoj Cj, stavlja poruku u svoj red čekanja, šalje odgovor(i,T(j)) procesu Pi
c))Pi smije u KO – kad je njegov zahtjev na početku reda, te kad je primio odgovore (od svih ostalih procesa) čije su vremenske oznake >T(i)
d) Pi izlazi iz KO – odstrani iz svog reda zahtjev(i,T(i)), pošalje svim ostalima izlazak(i,T(i)*) [!! T(i)* je vrijednost prvotno poslanog zahtjeva]
e) Pj primi izlazak(i,T(i)) – iz svog reda izbacuje zahtjev(i,T(i))

<u>3x(n-1)</u> poruka za in-out KO

4. Protokol R-A

Procesi šalju odgovore samo kad ustanove da ne žele u KO ili kad nemaju pravo prvenstva (to je bitna razlika od Lamporta, kaže daxxx da mu se n da opet ista pravila pisat ☺)

2x(n-1) poruka za in-out KO (učinkovitiji od Lamporta)

5. Lokalni i globalni logički sat

<u>Lokalni logički sat</u> Ci – brojilo, povećava vrijednost prilikom svakog karakterističnog događaja u Pi

- Jednoznačno utvrđivanje redoslijeda događaja unutar 1 procesa
- a →b ako Ci(a) < Ci(b)

<u>Globalni logički sat</u>

- kada Pi šalje poruku m, uz nju pridodaje vremensku oznaku Tm = Ci
- kada Pj primi m, postavlja svoj Cj na vrijednost veću od prethodne i veću od pristigle Tm
- a ==>b ako Ci(a)<Cj(b) || [Ci(a)=Cj(b)) && i<j]