

PREPREDVODNIK

- procesor i spremnik su povezani preko sabimica
- sklop koji s jedne strane prima MI, a s druge je spojen na sabimicu je DMA
- procesor je automat koji čeka prekid dok se nedogodi
- koji su registri DMA?
- koje strukture podataka koristi Deckerov/Tetensov model? 2 rezervice i prav
- tko sve može poslati signal procesu? jezgra, proces sam sebi, drugi proces, konzum
- kako se zove DMA na hr? neposredni pristup spremniku
- nabroji monitorске funkcije sa predavanja/lažosa
- koliko ima binarnih & općih za potrebe i proizvodnje

S K L O P O V L J E (2)

O. S.



O.S. - skup programa koji olakšavaju rad na računalu

- osnovna funkcija računala je da se na vjenče igra :)
- Windowsi nđ: 1. koji omogućuju više zadatak u rad.

2. MODEL JEDNOSTAVNOG RAČUNALA - VAN NEUMANNOV MODEL RAČUNALA

- REGISTRI :
 - adresni uređuregistar
 - podatkovni uređuregistar
 - instrukcijski registar
 - programsko brnje (PC)
 - registar kazaljke stoga (SP)
 - registar stanja (status registar)
 - opći registar (RD-Rk)

- RAD PROCESORA: ponavljati q

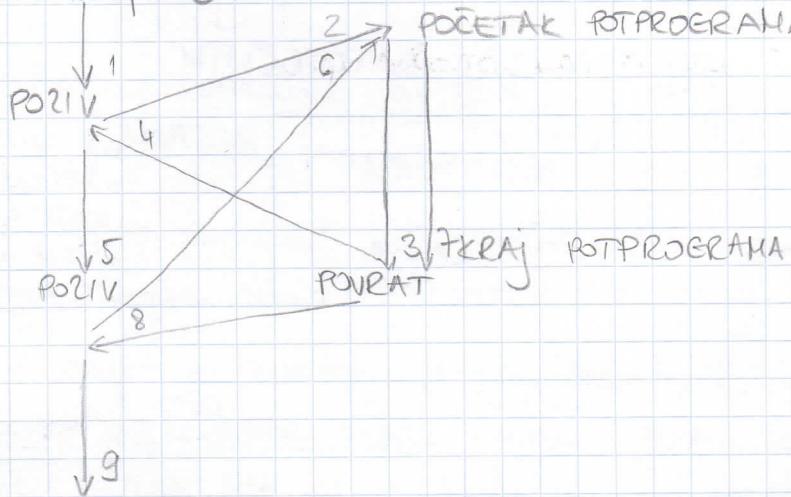
dohvatiti iz spremnika instrukciju na koju pokazuje prog. brojilo
 dekodirati instrukciju, odrediti operaciju koju treba izvesti
 posredovati sadržaj programskog brnja, takođe pokazuje sled. instrukciju
 bitrediti odluke dobiti operand i odje poslati u rezultat
 operande dovesti u ALU, izvesti zadatu operaciju
 pohraniti rezultat u oglediste

dok je (procesor uključen)

PROGRAM je staticki niz instrukcija

PROCES je skup radunalnih resursa koji nam omogućuju izvođenje programa

DRETVA (THREAD) je niz instrukcija u izvođenju, izvodi ono što piše segeti klapić



PODSETNIK (u rasp):

CALL BY VALUE (valo parametara)

CALL BY REFERENCE (puno parametara)

INSTRUKCIJA DRETNE

- u spremniku se mora rezervirati prostor gdje će se polaziti niz instrukcija, mora se rezervirati mjesto za podatke i stog gdje ćemo polaziti međucijelne i adrese
- radunalni proces mora imati NAJMANJE JEDNU DRETU
- u nekom trenutku može biti aktiven samo jedna dretva, ostale cekaju
- skakanje OS-a u procesor dok dretva neči radi (tipkovica digne zastavice: "je procesor imao nest za tebe (blink blink) pročitaj u moju podatkovnu registru.")

PITANJA:

- KONTEKST DRETNE - je sadržaj svih registara
- rad i ponasanje PROCESORA: instrukcije i registri
- višeprogramski rad: dretve - tako da radi jednu, pa drugu, pa treću itd..

3. PREKIDNI RAD

- registar se sastoji od skupa bistabila
- u našem strojnemu jeziku smo konzistit 6 instrukcija:

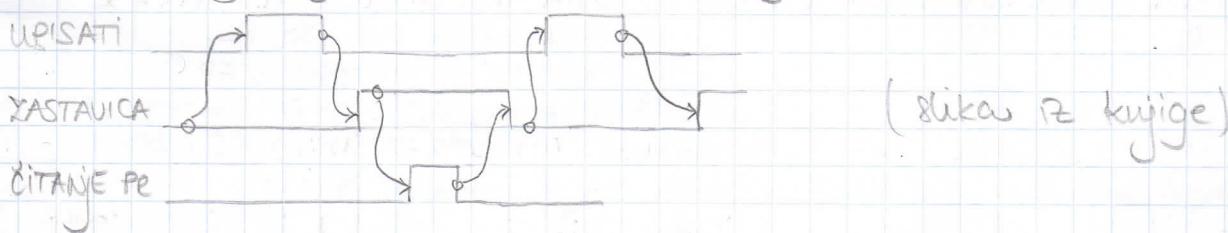
|
- LDR r0, [r1]
- STR r0, [r1]
- ADD r1, \$imm
- CMP r0, #0
- BEQ PETUA
- BAL PETUA

↓
Samo na ponavljanima

PSEUDOKOD:

dok je (zastavica == 0); // RADNO ČERANJE → procesor ispituje besomnje zastavici
čitej PR;

protokol dužeg nukovanja - funkcionira sa dve žice: UPISATI i ZASTAVICA
skidaju signal dužog nukovanja:



PR 3.1. → 0,1% vremena procesor radi koristan posao
rješuje problema u PREKIDNOM RADU

? ŠTO PROCESOR RADI KADA SE DOGODI PREKID? → 5 radnji

ako je (prekidni signal postavljen i omogućeno je prekidaće) {

- 1) omogući dalje prekide
- 2) i 3) prebaciti adresiranje u sustavski adresni prostor i aktivirati sustavsku kazaljku stoga
- 4) pohraniti programsko brojilo na sustavski stog
- 5) stvari u programsko brojilo adresu pop. za obradu prekida; }

? ŠTO SE DOGODI KADA OBRAĐA PREKIDA ZAVRŠI?

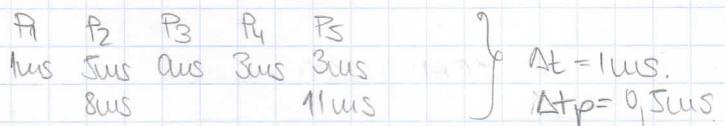
str. 40 u knjizi

PRIMJER 3.4 → na ponavljajuću signalu, možda ne regularnu

Zadatak 3.1. (M1)

javljaju se prekidi P₁, P₂, P₃, P₄ i P₅. Prekidi P₁, P₃, P₄ javljaju se u 1ms, 0ms i 3ms.

Prekid P₂ javlja se dva put u trenucima 5 i 8 ms, prekid P₅ javlja se dva put u trenucima 3 i 11 ms. P₅ ima najviši prioritet. Svakac od prekidnih procedura traje 1ms. Graficki prikazati GP, P_i i POPP. POPP traje 0,5ms.



a) u idealnom slučaju

b) bez sklopa za prihvatanje prekida

c) sa sklopm za prihvatanje prekida

a)



c) (nastavak)

(1) dogodio se prekid razine 3

(2) završila obrada prekida razine 3 i propustila se prekid razine 1

(3) završetak obrade prekida razine 1 i propuska se do procesora prekid razine 5, a P4 se pamti

(4) završetak obrade P5, propuska prekid razine 4 i pamti prekid razine 2

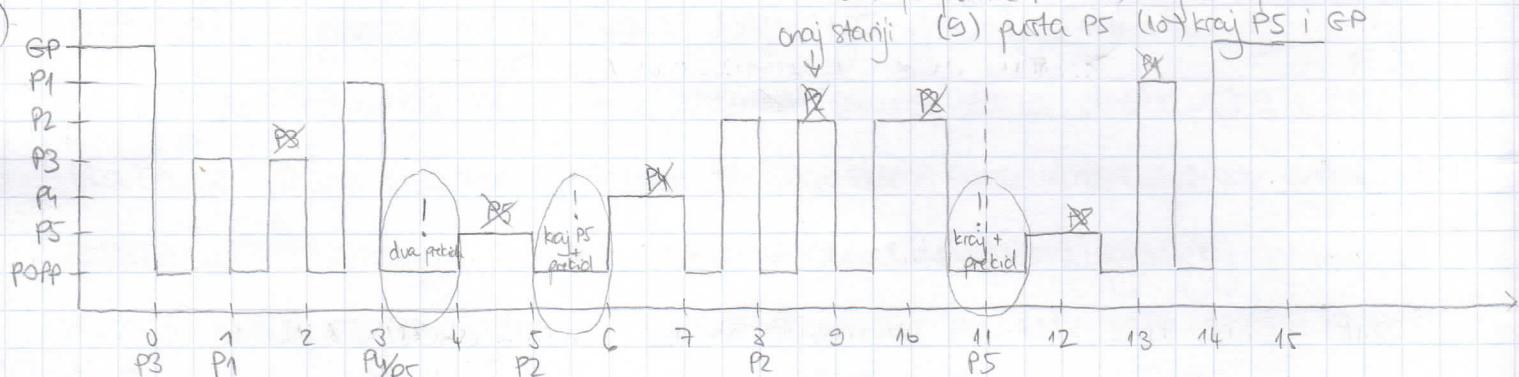
(5) završetak P4, propuska P2

(6) završetak P2 i povratak u GP

(7) propuska P2 (8) kraj P2 i povratak GP

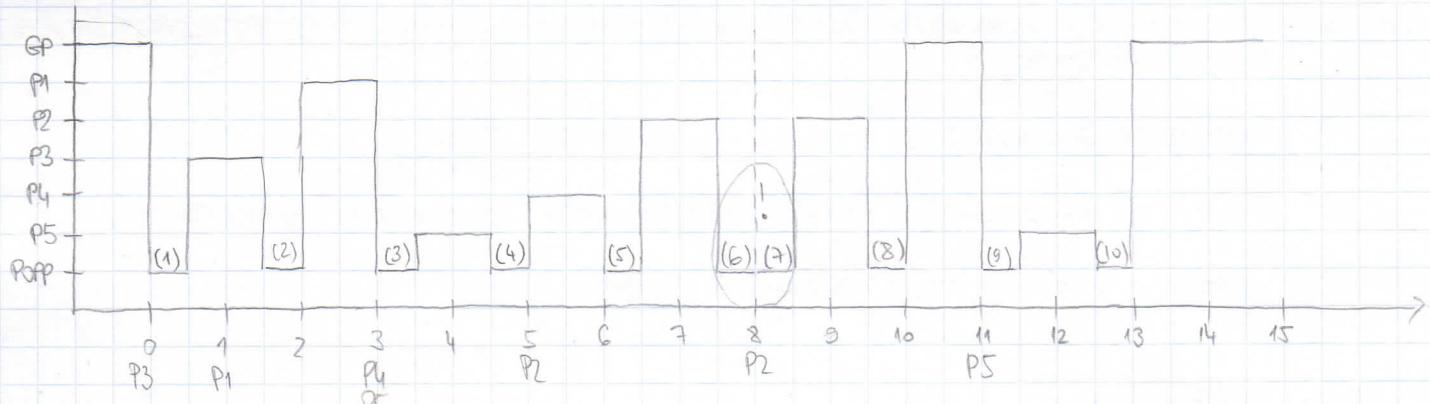
onaj stanji (9) putna P5 (10) kraj P5 i GP

b)



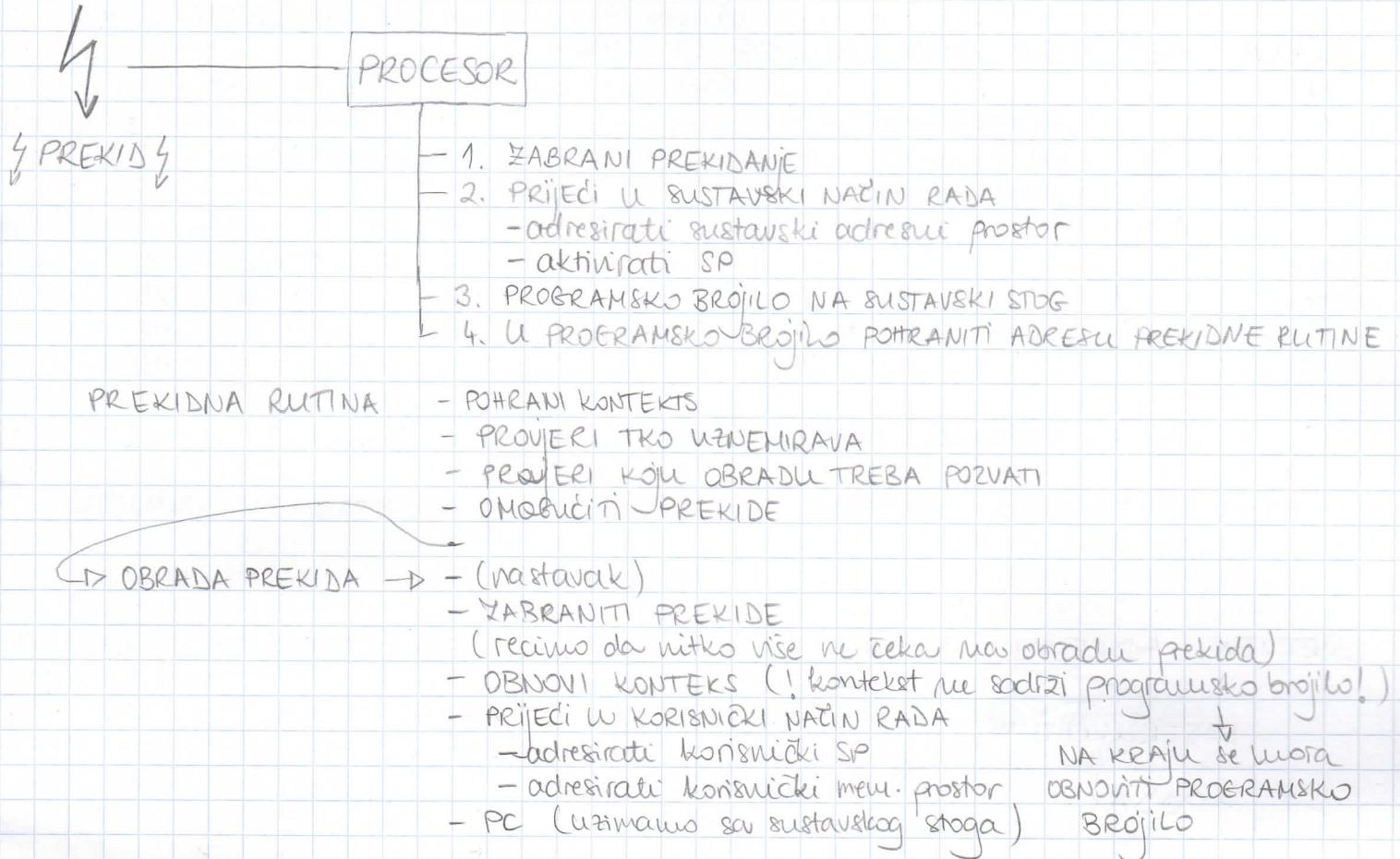
- u slučaju kada nemamo sklopa za prihvatanje prekida, POPP se poziva svaki put kada se dogodi prekid; svaki put kada obrada nekog prekida završi i svaki put kada se dogodi prekid veće razine od tečućeg prekida, (u ostalim slučajevima pamti te kontrolne tastanice)

c)



- u slučaju kada sklopovi sadrže i sklop za prihvatanje prekida POPP se poziva svaki put kada obrada nekog prekida završi i svaki put kada se dogodi prekid veće razine od tečućeg prekida, (u ostalim slučajevima pamti te kontrolne tastanice)

PONAVLJANJE:



$j = fp \rightarrow JEZGRINA\ FUNKCIJA$

- Na početku svake jezgrine funkcije pohranjuje kontekst

? Što je pristupni sklop?

? Što znači pohraniti kontekst?

? Koje registre imaju DMA i pristupni sklop i čemu registri služe?

? Što DMA sklop radi?

Dretva i }

N. K. O. ; - nekritični dio (nije mogući klub Osijek je)

K. O. ; - kritični dio

↓

NE smiju biti dretve u N. K. O.

Dretva j }

N. K. O.

K. O.

}

D.Z. U sustavu imate 10 dretvi koji broje do milijun. Kolika je najmanja moguća vrijednost, nakon što sve dretve izvrše svog posao, ako je početna vrijednost varijable $A=0$.

ODE: 2

nije bitan broj dretvi i do koliko broji



CIKLIČKA DRETVNA

dretva i {

dok je (1) {

N.K.O.

K.O.

}

osnovni zahtjev: međusobno isključujuće

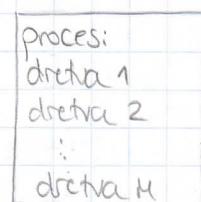
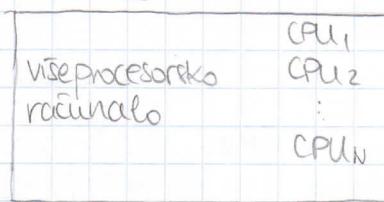
dodatni zahtjev: 1. algoritam funkcioniра i onda kada su brzine izvršenja dretvi razlike

2. ako neka dretva zastane u N.K.O. to ne smije spriječiti drugu dretvu da uđe u K.O.

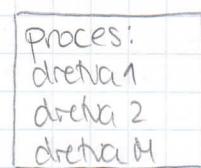
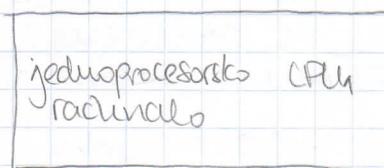
3. izbor koja dretva ulazi u K.O. mora se obaviti u konacnom vremenu (vrlo kratkom vremenu)

PONAVLJANJE:

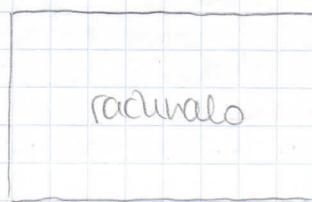
OSNOVNI pojmovi: višedretveni, višezasadaci, višekorisnici rad



višedretveni rad
istovremeno izvršenje
dretvi na razlicitim
procesorima



višedretveni rad
naizmjenično izvršuje
dretvi



Proces 1
'n' dretvi, korisnik 'p'



Proces M
'z' dretvi, korisnik 'r'



višedretveni rad

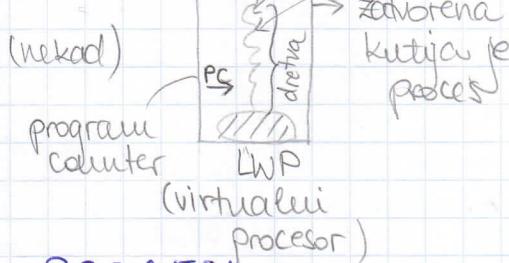


višezasadaci rad

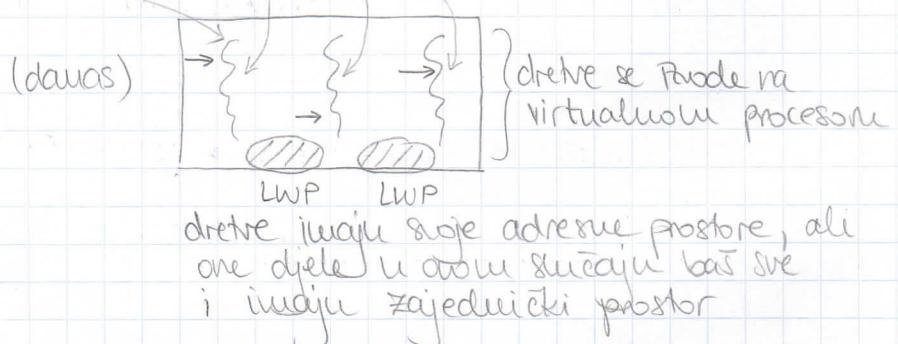
višekorisnici rad

Priprema za pisanju za 3. ježku:

PROCESI

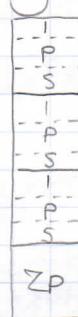


DRETE



- proces se sastoji od BAREM jedne dretve
- ? kako napraviti program koji stvara više procesa?
- proces koji stvara novi proces naziva se **RODITEJ**, a novostvoreni se nazivaju proces **DIJETE**
- ! PROCES DIJETE JE KOPIJA RODITEJA

I	(instrukcije)
P	(podatke)
S	(stog)
ZP	(zajednički prostor)



- proces dijete i proces roditelj NEAJU NIŠTA ZAJEDNIČKO
 - ↳ tako su kod roditelja deklarirali globalnu varijablu a dijete će tako imati varijablu → kada kod dijeteta se prosljedi vrijednost on kod roditelja se neće prenijeti
- u programskom jeziku C GLOBALNE varijable su zajedničke za sve dretve, a LOKALNE su rezane uz određenu dretvu
- u pravilu proces dijete mora završiti PRIJE roditelja
- **ORPHAN** - proces bez roditelja, sruče (NE može biti na računalu)
 - os mu je skrbnik
- **ZOMBIE** - završio je s procesom, ali u tablici stavlja još nije pobrisan
- **INT FORK (void)** - služi za stvaranje novog procesa
 - poziva je roditelj
 - rezultat funkcije su RODITEJ i DIJETE

↳ dijete dobiva 0

↳ roditelj dobiva ≠ 0 i to je PID dijeteta

↳ -1 → dijete nije snopeno

Kako snimiti sustav?

while (1) fork(); ← NE RADITI !!

↳ on u beskonačnosti stvara nove procese

DRETVE

- thread_create (...) - svaka napravio svoje
- pthread_create (...) - ovo radiju na labosima
↳ u windowsima napravili create_thread (...)
- SPECIFIČNO ZA DRETVU:

- Svaka dretva ima svoj TID
- stavlja registara
- kaznica stoga
- stog
- zastavice
- prioriteti
- privatni prostor svake dretve

ZADATAK: napisati program koji stvara tri procesa kroz dretve gdje svaka/i dretva/ proces ispisuje 5 puta "Ja sam proces/dretva,
ali ovo je proces i"

PROCESI

```
#include <stdio.h>
void Proces (int i){
    int j;
    for(j=1; j<6; j++){
        printf("Proces: %d, prolaz %d\n", i, j);
        sleep(1);
    }
}
```

```
int main (void) {
    int i;
    for(i=0; i<3; i++){
        switch(fork()){
            case -1: printf("ne ide"); exit(1);
            case 0: Proces(i); exit(0);
            default: break // nastavi posao roditelja
        }
    }
}
```

```
for (i=0; i<3; i++) wait(NULL);
return 0;
```

↓ reakcije milotoke
da nema ore li nije
nastala bi 3 ofayka
↓
pojavljuje se
procesa

DRETVE

```
#include < stdio.h >
#include < pthread.h >
void *Dretva (void *rbr){
    int j;
    for(j=1; j<6; j++){
        printf("Dretva %d prolaz %d", *((int *)rbr), j);
        sleep(1);
    }
}
```

```
int main (void)
{
    int i; BR[3];
    (struct) pthread_t t[3];
    for (i=0; i<3; i++){
        BR[i] = i; // upisuju se podaci
        if (pthread_create (&t[i], NULL, Dretva, (void *) &BR[i])) {
            printf("Ne ide\n");
            exit(1);
        }
    }
}
```

```
for (i=0; i<3; i++) pthread_join (t[i], NULL);
return 0;
```

↓ pre nego sto bi
poceo ja bi zavrsio
processa
→ vez ore li nije
nista
↓ ne bi ispisao nista
unutar processa

RAZLIKA U PONAŠANJU DEKKEROVOG I PETERSONOVOG ALGORITMA

DEKKEROV (moramo znati)

dok je (1) {

$Z[1] = 1;$

dok je ($Z[j] \neq 0$) {

ovo je isto
akao je ($PRAVO \neq 1$) {

$Z[1] = 0;$

dok je ($PRAVO \neq 1$);

$Z[1] = 1;$

}

K.O.

$PRAVO = j;$

$Z[1] = 0;$

N.K.O.

}

- dretva koja duže vremena nije bila u K.O. ima prednost
- ovisi o početnoj vrijednosti varijable **PRAVO**

PETERSONOV (nemoramo znati)

dok je (1) {

$Z[1] = 1;$

$PRAVO = j;$

dok je ($(PRAVO == j) \wedge (Z[j] == 1)$);

K.O. j

$Z[1] = 0;$

N.K.O;

}

- daje preduost drugoj dretvi

- ne ovisi o početnoj vrijednosti varijable **PRAVO**

LAMPORTOV - str 83. u kujizi (moramo ga znati za ispit)

dok je (1) {

$MAZ[1] = 1;$

citaj $ZADNJI_BR;$

$Broj[1] = ZADNJI_BR;$

$ZADNJI_BR = BROJEI[1];$

$MAZ[1] = 0;$

$\forall a (j = 0; j < N; j++) {$

 citaj $MAZ[j];$

 dok je ($MAZ[j] \neq 1$) { citaj $MAZ[j];$ }

 citaj $BROJ[j];$

 dok je ($(BROJ[j] \neq 0) \wedge ((BROJ[j], j) < (BROJ[i], i))$) { citaj $BROJ[j];$ }

K.O.

$BROJ[1] = 0;$

N.K.O

}

NEDJEJIVE INSTRUKCIJE:

- TAS (test and set - ispitati i postaviti)
- SWAP (swap - zamjeniti)
- FETCH & ADD

dok je (1) {

dok je (TAS(ZASTAVICA) == 1); // RADNO ČEKANJE

K.O.;

 → FALSE je samo onda ako je zastavica bila 0, i zastavica postaje 1

ZASTAVICA = 0;

N.K.O.;

}

- u pseudokodu i u mnemonickom obliku moramo znati sreću za ispit

5. JEZGRA OPERACIJSKOG SUSTAVA

KONTEKST - sadržaj svih registara

- Moramo dretni zapamtiti gdje je stala, dokle pautilis joj kontekst
- JEZGRINE FUNKCIJE rade s kontekstom

JEZGRA OPERACIJSKOG SUSTAVA sastoji se od:

- 1. struktura podataka jezgre
- 2. JFJA (jezgrinih funkcija)

- u jezgri jezgrine funkcije bave se strukturama podataka
- ↳ - u jezgri uklazimo kada se dogodi prekid → poziv jezgrine funkcije

Postoje 3 VRSTE PREKIDA

1. PROGRAMSKI PREKIDI (može razvati bilo koja dretva)
2. SKLOPOVUSKI PREKIDI (razina u nafra - u operaciju pokreće jezgra)
3. PREKID OD SATA (ne svaki put nego višekratnik, narušava sata kada neke dretve odgadaju - da znaju kolko dretva spava)

ULAZAK u JEZGRU zbiva se pod utjecajem prekida, a svodi se na poziv neke jezgrine funkcije

IZLAZAK iz JEZGRE svodi se na pokretanje jedne od pripravnih dretvi (pri čemu procesor prelazi u busnički modus rada)

STRUKTURE PODATAKA : 1. OPISNIK dretve
(OSNOVNE) 2. opisući se nalaze u LISTAMA (dretva čeka)

- jezgrine funkcije rade s LISTAMA i s OPISNICIMA dretve

- ! SADRŽAJ opisnika dretve :
- 1. katalike (najmanje 1)
 - 2. identifikator procesa (PID)
 - 3. identifikator dretve (TID)
 - 4. stanje dretve
 - 5. prioritet dretve
 - 6. porečna adresa dretvenog adresnog prostora
 - 7. veličina dretvenog adresnog prostora
 - 8. adresa prve instancije dretve
 - 9. zadano korištenje
 - 10. prostor za snimanje konteksta dretve
 - n. programsko brojilo (PC)

- LISTE :**
- LISTA POSNOVNIH dretvi → Postojeće - D
 - AKTIVNA dretva (samo 1!) → Aktivna - D (zapravo nije lista)
 - FIFO (first in first out)
 - PRIORITETNA (stavlja po važnosti na mjesto)
- } red pripravnih dretvi

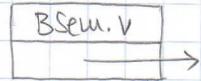
BLOKIRANA dretva: čeka na binarnou semafore
 čeka na opću semafore
 čeka na istek zadalog intervala kašnjenja
 čeka na završetak neke operacije

BINARNI SEMAFOR

→ je jezgrin mehanizam koji služi za sinkronizaciju dretve

STRUKTURA PODATAKA:

- jedne varijable : BSem.v
- jedne kozaljke



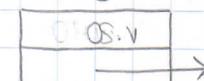
- | | | |
|-------------------------|---|----------------------|
| 0 crveno
(ne prolaz) | ! | 1 zeleno
(prolaz) |
|-------------------------|---|----------------------|

OPĆI SEMAFOR

→ je jezgrin mehanizam koji služi za sinkronizaciju dretve

STRUKTURA PODATAKA:

- jedna varijabla OSem.v
- jedne kozaljke



- | | | |
|----------------------|---|----------------------|
| 0 Crveno
[-oo, 0] | ! | 1 zeleno
[1, +oo) |
|----------------------|---|----------------------|
- funtacija

RED se formira po redu prijedela

moguća STANJA:

- prolazan
- neprolazan i nema nikoga u redu
- neprolazan i ima nekoga u redu

JEZGRINE FUNKCIJE: → ispitej_BSEM(i)
 → postavi_BSEM(i)

RED se formira po redu prisjeda

moguća STANJA:

- prolazan
- neprolazan i nema nikoga na redu
- prolazan i ima nekoga na redu

JEZGRINE FUNKCIJE: → ispitaj_OSCI(i)
 → postavi_OSCI(i)

K.O. → ako je više dretvi na zebri

ODGOĐENDE DRETVE → str. 99

MOGUĆA STANJA DRETVI su: (krugovi na slici)

- 1.) blokirana (ima ih 4)
- 2.) aktivna
- 3.) pripravna
- 4.) pasivne

- strelice su 1-fje

misao je znati nacrtati za MI

j-Fje

ULAZ: pokraniti kontekst u opisnik Aktivna_D

AKTIVIRANJE: preuijetiti prvi opisnik iz reda Pripravne_D u red Aktivna_D
obnoviti kontekst iz opisnika Aktivna_D
omoguciti prekidače
vratiti se iz prekidnog radnina

IZLAZ: aktivirati prvu dretvu iz reda Pripravna_D

FUNKCIJE ZA BINARNI SEMAFOR:

Ispitati_Bsem(1)
K.O.

Postaviti_Bsem(1)
N.K.O.

BINARNI SEMAFOR

Ispitati_Bsem(1) {

pokraniti kontekst u opisnik Aktivna_D;

ako je (Bsem[1].v == 1) { | ako je zeleno |

Bsem[1].v = 0;

obnoviti kontekst iz opisnika Aktivna_D;

omoguciti prekidače;

vratiti se iz prekidnog radnina;

}

inace { | ako je crveno |

preuijetiti opisnik iz reda Aktivna_D
u red Bsem[1];

aktivirati prvu dretvu iz reda Pripravne_D; aktivirati prvu dretvu iz reda Pripravne_D;

}

Postaviti_Bsem(1) {

pokraniti kontekst u opisnik Aktivna_D;

preuijetiti opisnik iz reda Aktivna_D u

Pripravne_D;

ako je (red Bsem[1] neprazan) {

preuijetiti prvi opisnik iz reda Bsem[1]
u red Pripravne_D;

}

inace {

Bsem[1].v = 1;

}

aktivirati prvu dretvu iz reda Pripravne_D;

OPCI SEMAFOR

Ispitati_Os(j) {

pokraniti kontekst u opisnik Aktivna_D;
Os[j].v = Os[j].v - 1;

ako je (Os[j].v >= 0) {

obnoviti kontekst iz opisnika Aktivna_D;

omoguciti prekidače;

vratiti se iz prekidnog radnina;

}

inace {

preuijetiti opisnik iz reda Aktivna_D u
red Os[j];

Postaviti_Os(j) {

pokraniti kontekst u opisnik Aktivna_D;

preuijetiti opisnik iz reda Aktivna_D u

Pripravne_D;

Os[j].v = Os.v + 1;

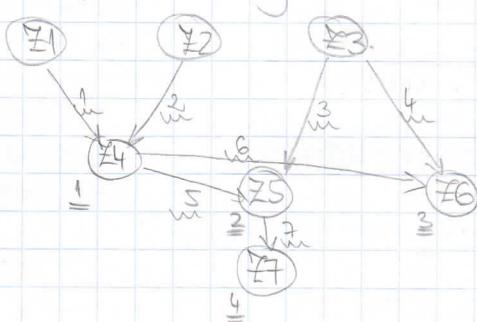
}

Zad 5.1.: Za ostvarenije sustava dreti prema slici (vj. zad. 4.1.) koniste se binarni (XOR) opći semafor.

a) koliko semafora se potrebno za sinkronizaciju?

b) koje su početne vrijednosti?

slika:



BINARNI SEMAFOR:

a) broj semafora = broj strelica

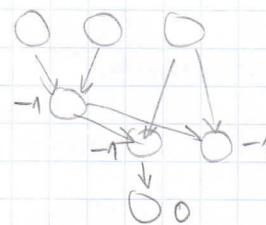
potrebno je 7 semafora; $\#Bsem = \#\downarrow = 7$

b) $Bsem(i).v = 0$;

OPĆI SEMAFOR

a) broj semafora = broj kružica koji primaju strelice u sebe
potrebno je 4 semafora; $\#OS = \#\nearrow = 4$

b) $OS(i).v = -(\#\nearrow - 1)$



c) Neka je T_i tekst zadatka i. Prosinti tekst u T'_i s minimalnim brojem poziva j-fja (čekaj postavi $Bsem, OS$)

BINARNI SEMAFOR

$T_i \rightarrow T'_i$

$T_1': T_1$
postavi $Bsem(1)$
 $T_2': T_2$
postavi $Bsem(2)$
(...)
 $T_4':$
čekaj $Bsem(1)$
čekaj $Bsem(2)$
 T_4
postavi $Bsem(5)$
postavi $Bsem(6)$

OPĆI SEMAFOR

$T_1': T_1$
postavi $OS(1)$
 $T_2': T_2$
postavi $OS(1)$
(...)
 $T_4':$
čekaj $OS(1)$
 T_4
postavi $OS(2)$
postavi $OS(3)$

* za svaku koja ide u kružicu je čekaj
za svaku strelicu koja ide iz kružice je postavi

Zad 5.2: Synchronizacija 3 dretve uz pomoć binarnog semafora!

Uzna - D {

```
dok je (1) {
    čekaj podatke;
    => čekaj_BSEM(1);
    salji podatke;
    => postaviti_BSEM(2)
}
```

Radna - D {

```
dok je (1) {
    => čekaj_BSEM(2)
    priuđi podatke;
    => postaviti_BSEM(1)
    obradi podatke;
    => čekaj_BSEM(3)
    salji podatke;
    => postaviti_BSEM(4)
}
```

Izlazna - D {

```
dok je (1) {
    => čekaj_BSEM(4)
    priuđi podatke;
    => postaviti_BSEM(3)
    ispiši;
}
```

Početne vrijednosti

$$BSEM(1).v = 1$$

$$BSEM(2).v = 0$$

$$BSEM(3).v = 1 \text{ (zato što nema mesta u unutra)}$$

$$BSEM(4).v = 0$$

Prve dve: Uzna - Radna
druge dve: Radna - Izlazna

Brojački opći semafor - Osem

STRUKTURA PODATAKA:

- jedna varijable: Osem.v
- jedna kazaljka

Osem.v

0 = crveno

1 = zeleno (max.)

JEZGRINE FUNKCIJE: Ispitati_Osem(j)
Postaviti_Osem(j)

Ispitati_Osem(j) {

pohraniti kontekst u opisnik Akt.-D;
ako je (Osem[j].v >= 1) {
 Osem[j].v = Osem[j].v - 1;
 obnoviti kontekst iz opisnika Akt.-D;
 mogući prekidanje;
 vratiti se iz prekidačnog macina;

inace {

prenijestiti opisnik iz reda Akt.-D
 u red Osem[j];

aktivirati prvu dretvu iz reda
 Pripravne-D;

}

Postaviti_Osem(j) {

pohraniti kontekst u opisnik Akt.-D;
 prenijestiti opisnik iz reda Akt.-D u
 red Pripravne-D;
ako je (red Osem(j) neprazan) {
 prenijestiti pri opisnik iz reda Osem(j)
 u red Pripravne-D;

inace {

Osem[j].v = Osem[j].v + 1;

aktivirati prvu dretvu iz reda Pripravne-D

REŽIME SEMAFORA

SEMAFOR	ZELENO	CRVENO	
		(nema nikoga u redu)	(ima nekoga u redu)
BINARNI OPĆI	V=0 (postavi crveno) čekaj_Bsem proti semafor		premijestiti u red BSEM
	V=1 ili nista	V=1 (postavi zeleno)	premijestiti prvu dretvu iz reda BSEM u red pripravne-D
OPĆI ZEKAT POZ	V--; čekaj_OS proti semafor	V--; premijestiti u red OS	
	V++; Postavi_OS	V++; premijestiti u red OSEM	V++; ako je $(V=0)$ premijestiti prvu dretvu iz reda OS u red pripravne-D
OPĆI ZEKAT POZ	V--; čekaj_Osem proti semafor	V--;	premijestiti prvu dretvu iz reda OSEM u red pripravne-D
	V++; Postavi_Osem	V++;	premijestiti prvu dretvu iz reda Osem u red pripravne-D

Zad 5.3. Razvili smo sustav s tri dretve pušača i jednu dretvu trgovca. Svaki pušač neprestano savija cigarete i puši. Kako bi se savila i popustila cigareta potrebno je imati tri sastojka: papir, duhan i šibice (bez filtera :)). Jedan pušač ima u neograničenim kolicinama samo papir, treci samo duhan, treci samo šibice. Trgovac ima sva tri sastojka u neograničenim kolicinama. Trgovac masovno stavljaju na stol dva razlicita sastojka. Pušač koji ima treci sastojak signalizira trgovcu, savija cigaretu i puši. Trgovac stavlja nova dva sastojka na stol; ciklus se ponavlja. Na početku, stol je prazan. Napisati dretve pušaća i trgovca tako da se one međusobno ispravno sinkroniziraju uz ponud binarnih semafora. Napisati početne inicijalnosti semafora.
 (!to nije reklama za cigarete!)

→ Petak 1971. g.

- ISPRAVNO SINKRONIZIRATI : 1) redoslijed izvođenja mora biti ispravan
 2) neka sredstva treba konstituti međusobno isključivo
 3) nema radnog čekanja
 4) nema beskonačne petlje
 5) nema potpunog zastoja

I sjesije: → koristimo dva binarna semafora

NEOGRAĐENE KOLIČINE → dok je (1)

NASUMICE → random

SIGNALIZIRA → postavi binarni semafor

STOL JE PRAZAN → definiraće početnu vrijednost (!, okarezno ih pisati)

Dretva_Traovac {
 dok je (1) {

čekaj_BSEM (stol prazan);
 stavi_dva_sastojka_na_stol;
 postavi_BSEM (ima_nesto); } }

Dretva_Pušač {
 dok je (1) {
 čekaj_BSEM (ima_nesto);
 ako_xu_postavljeni_sastojci_koji_nam_trebatu {
 uzvi_sastojke;
 postavi_BSEM (stol_prazan);
 skocaj_i_pusi; } }
 inace {
 postavi_BSEM (ima_nesto); } }

POCETNE VRJEDNOSTI:

BSEM (stol prazan) - V=1
 BSEM (ima_nesto) - V=0

II sjesije:

PONAVLJANJE 1. CIKLUSA

5.5.5.

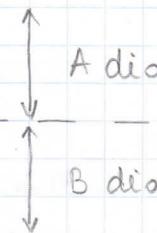
Zad 5.4. Prioritetni redovi i semafori

U jednoprocesorskom računalu pokrenut je sustav dreti D_1, D_2, D_3, D_4 s prioritetima 1, 2, 3, 4. Najviši prioritet je 4. Svi zadaci koji obavljaju dretre su u log oblika Dx . Red pripravljenih dreti i red sematofora su prioritetu. Aktivna je ona dretva koja je prva u redu pripravljenih (nešta posebnu reda aktivnih dreti). Prije pokretanja svake dretvi semafor S je zatvoren. Nakon nekog vremena sve dretve se nađu u redu sematofora S. Ako se tada pozove procedura Postani_Bsem, što će se ispisati na zaslonu?

Dretva $Dx \{$

dok je (1) {

čekaj_Bsem();
piji(Px);
Postani_Bsem();
pisi(Zx);



→ Ako ih u nekogu u redu onda preusti;
ako nešta postavi semafor!

2j:

red Bsem	r	red Pripravna_D	P	Z
4 3 2 1	0	-	4	
3 2 1	0	4 a	4	
2 1	0	4 b 3 a	4	
4 2 1	0	3 a	3	
2 1	0	4 a 3 b	4	
1	0	4 b 3 b 2 a	4	
4 1	0	3 b 2 a	3	
4 3 1	0	2 a	2	
3 1	0	4 a 2 b	4	
1	0	4 b 3 a 2 b	4	
4 1	0	3 a 2 b	3	
1	0	4 a 3 b 2 b	4	
-	0	4 b 3 b 2 b 1 a	4	
4	0	3 b 2 b 1 a	3	
4 3	0	2 b 1 a	2	
4 3 2	0	1 a	1	
3 2	0	4 a 1 b	4	
2	0	4 b 3 a 1 b	4	
4 2	0	3 a 1 b	3	
2	0	4 a 3 b 1 b	4	
-	0	4 b 3 b 2 a 1 b	4	
4	0	3 b 2 a 1 b	3	
4 3	0	2 a 1 b	2	
3	0	4 a 2 b 1 b	4	
-	0	4 b 3 a 2 b 1 b	4	
4	0	3 a 2 b 1 b	3	

red Bsem	r	red Pripravna_D	P	Z
-	0	4 a 3 b 2 b 1 b	4	
-	1	4 b 3 b 2 b 1 b	4	
0	4 a 3 b 2 b 1 b	4		

→ beskonačna petlja

→ sa vjerojatnošću 1 održav ispit
Idolazi nov ispit!

... sa for petljom ili 5 dretvi...

6. MEDIJ DRETVENA KOMUNIKACIJA I KONCEPCIJA MONITORA

PROIZVODAČ je ciklička dretva u kojoj se generiraju podaci, ti se podaci šaliju u obliku ponuke i šalju potrošaču

POTROŠAČ je ciklička dretva koja čeka na ponuke, prihvata ponuku i zatim podatke prispeče u ponuci potroši

- proizvodač i potrošač odnijaju svoj posao proizvodnjom brzine
- ponuke treba poslati kako bi se proizvodač u trenutku intervalu sukladno proizvodi ponuka mogao potrošiti
- za potraživanje ponuka potrebit će račun meduspremnik
- slijediti u spremnik prostor procesa kogom pripadaju obje nože dretve

- meduspremnik \rightarrow MS (neograničeni spremnik)

- indeks UL pokazuje na prazni pretinac u koji potrošač treba slijediti sljedeći ponuku

- indeks iz koji pokazuje na pretinac iz kojeg proizvodač upravo treba pribuzeti ponuku

\rightarrow brojavi ponuka u spremniku mogao poslužiti BROJACKI SEMAFOR Osim

RED ČEKANJA zajedno s potrošačem koji možemo nazvati poslužiteljem (eng. server) čini najjednostavniji model za analizu sustava.

ograničeni meduspremnik (u realnosti) \rightarrow veličina MS = N pretinaca

funkcija mod N \rightarrow N nije dobro jer su pretinici indeksirani od 0!

- sljedeći problem: više proizvodača, a jedan potrošač \rightarrow zastita varijable u \rightarrow sveu

? Kako rješiti problem proizvodača i potrošača preko meduspremnika?

\rightarrow Zopća i 2 binarna semafora \rightarrow više proizvodača i nje potrošača

\rightarrow za 1 potrošač i 1 proizvodač \rightarrow samo 2 opća

\rightarrow za neograničeni spremnik \rightarrow samo 1 opći

SKLADIŠTE PRETINACA

Zad. 6.1.

Modelirati vrtuljak s dva tipa dretvi: dretve posjetitelji i jednu dretvu vrtuljak. Dretvama posjetitelj se ne smije dozvoliti ukrcati na vrtuljak kada nije nema praznih mjesteta. Pokrenuti vrtuljak tek kada

je pun. Za sinkronizaciju konstitui opće semafore.
(verzija sa vanjskom binarnim semafornom)

↳ broj - slobodnih - mesta

Dretva Posjetitelj {

čekaj - Osem (vrtuljak)

sjedui i uživaj u vožnji

postaviti - Osem (sjeo)

čekaj - Osem (kraj);

opcionalo → postavi - Osem (izasao),

Dretva Vrtuljak {

dok je (!) {

za (i = 1 do br - mesta) {
postaviti - Osem (vrtuljak); } }

za (i = 1 do br - mesta) {
čekaj - Osem (sjeo); }
vrati vrtuljak;

za (i = 1 do br - mesta) {
postavi - Osem (kraj); }
opcionalo → za (i = 1 do br - mesta) {
čekaj - Osem (izasao); } }

NE PISATI : Osem (v) = Br - mesta !!! → to se u konzoli okolo gještu ne može napraviti!

Početne vrijednosti : Osem (vrtuljak) . v = 0 (o tome se brine vrtuljak)

Osem (sjeo) . v = 0

Osem (kraj) . v = 0

Osem (izasao) . v = 0

petje su vaporne!

Problemi sa semaforima:

① svaki semafor ispituje jedan mjesto

② ispitivanje semafora povezano je s zauzećem sredstva kojeg semafor stiti.
posljedica toga je: nije moguće obaviti više ispitivanja (par tek onda
zauzeti sredstvo)

③ potpuni ~~zastoj~~!

④ svako ispitivanje ili postavljanje semafora je poziv jedne funkcije
(a to dobro si prirodo kucavistog posla)

Potpuni zastoj

dretva D₁

Ispitaj - Bsem (K)
Ispitaj - Bsem (L)

Postani - Bsem (K)
Postani - Bsem (L)

dretva D₂

Ispitaj - Bsem (L)
Ispitaj - Bsem (K)

Postani - Bsem (L)
Postani - Bsem (K)

Početna stanja : Bsem (K). v = 1 (zelena)
Bsem (L). v = 1 (zelena)

PROBLEM PET FILOZOFA

Nužni uvjeti za nastajanje potpunog zastoja

- ① sredstva se koriste međusobno isključivo
- ② niko ne može dreti oduzeti sredstvo koje je ona zauzela
- ③ dretva drži jedno sredstvo dok čeka na dodjelu drugog sredstva

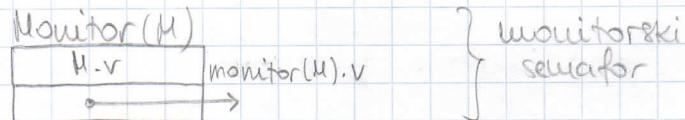
MONITOR je jezgrni mehanizam koji sluji za sinkronizaciju dretvi

monitor sastoji se od: monitorški funkcija i strukture podataka monitora

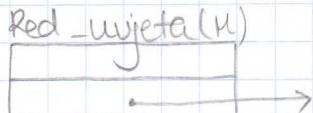
M-fa → Nije jezgrna fja, ora je konstrukta fja

↳ svaka mora biti zastitena binarnim semaforom, ali posebni monitorški semafor

STRUKTURA PODATAKA:



i određen broj redova ujeta:



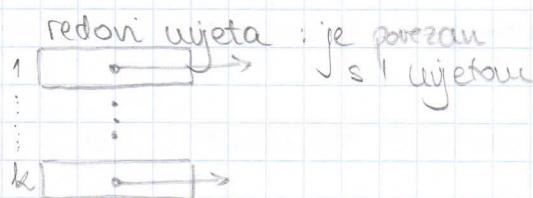
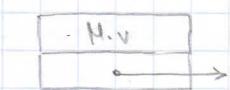
? monitorški
} semafor

Fje ZA OSTVARENJE MONITORA:

- ① ući_u_monitor (M)
- ② izaci_it_monitora (M)
- ③ uvrstiti_u_red_ujeta (M, k)
- ④ osloboediti_re_reda_ujeta (M, k)

Ponavljajuće:

MONITOR je jezgrin mehanizam koji služi za sinkronizaciju dretvi, a sastoji se od jezgrivih funkcija za stvaranje monitora i struktura podataka koje čine: struktura podataka monitorskog semafora i prizvoljani broj (učinjuće!) redova uijeta.



monitorske funkcije koniste jezgrine funkcije za stvaranje monitora

TEORIJA

- 1) uti u monitor (H)
- 2) Prodji u monitora (H)
- 3) Uvrsti u red uijeta (H,k)
- 4) Oslabodi iz reda uijeta (H,k)
u sebi učinju izaci-it-monitora

LABOSI

- mutex-lock (8M)
- mutex-unlock (8M)
- cond-wait (2M,K)
- cond-signal (2M,K)
- cond-broadcast (8M,K)

oslobodi sve!

svišao ovog
je zaključati
u monitor

j-funkcija cond-signal (H,I) {

pohraniti kontekst u opisnik Aktivne-D;

ako je (Red-ujeta (H,I) neprazan) {

premjestiti pri opisnik iz reda Red-ujeta [H,I] u red pripadajućeg Monitor(H)

okončati kontekst u opisnika Aktivna-D;

uvući prekidanje.

Vratiti se iz prekidačnog načina;
!

(neuva izaci-it-monitora)

PRAVILA KORISNIČKIH N-fja

- 1 U-fja je zaštićena monitorskim sekvencama
 - 2 Unijet se ne ispituje s Ako je nego se ispituje s Dok je
 - 3 natom uvrsti u red unijeta mora pisati vodi-u-monitor (ukoliko ili cond-wait ili mutex-lock i počne kraj monitorске funkcije)
 - 4! završetak (monitorска функција postigne k.o.) monitorске функције
↳ oslobođi ili zadrži
→ potencijalna opasnost da su dve direkcie (transfere) u procedu

Zad 6.2.: print: „prva
druga“

4

Drewno 1 f
wci w monitor (M);
ispisit ("prva");
prva = 1;
oslobodi 2. reda - wjeta (druga, M);

(B)

$\rightarrow !$ cond_signal (druga, M);
 mutex_unlock (M);

```

Drena 2 {
    wci u monitor( $\mu$ );
    dok je ( $\rho_{na} == 0$ ) {
        wroste u red_wijeta(dniga, $\mu$ );
        wci w monitor( $\mu$ );
    }
    ispis ("dniga");
    izacti z monitora( $\mu$ );
}

```

Dretnva 2 je

```

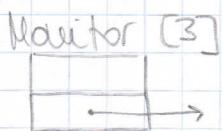
    mutex - lock(&M);
    dok je (PNA == 0) {
        cond - wait (dniga, M);
    }
    → ! mutex / lock (&A); → pravilo 3
    ?                                         NE vnesi
    ispisí ("dniga");
    mutex - unlock (&M);
}

```

- cond.-signal prenosi se iz reda prijeda u red monitorskog semafora, a ne kao oslobodilac reda prijeda koja prenosi se u red pripravnih
 - cond.-signal ne može izlaziti iz monitora
 - cond.-signal se pobrane za dretvu i zato NE VRIJEDI PRAVILA 3!

! UVRST - U - RED - WIJETA(X,M); = COND - WAIT(X,M)
 • UDI - U - MONITOR

Zad 6.3. problem 5 filozofa



K.O. → jesti

m-funkcije: Uzeti stupice
Spušti stupice

Zad 6.4. PROBLEM STAROG MOSTA

Star most je uski most zbog čega predstavlja ograničenje za promet.

Na njemu smiju biti najviše 3 auta koja voze u istom smjeru simulirati automobile zadavaju dretvom auto. Napisati u pseudokodu tražene monitorске funkcije.

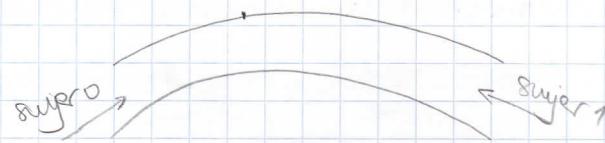
DretvoAuto (sujer) {

Popui - se - na - most (sujer);

Prijesti - most (); K.O.

Sidi - sa - mosta ();

}



I ulja Popui - se - na - most (sujer) {

Vodi - u - monitor (M);

dok je (most pun ili auto se voze u krovnu sujeru) {

čekaj u redu ujeta (M, 1);

} (kreni na most)

Pozadi - P - monitora (M);

I ulja Sidi - sa - mosta () {

Vodi - u - monitor (M);

Okončati sve logi čekaju u redu ujeta

i odvezi se s mosta;

Izadi - P - monitora;

→ za ispitivanje da li je most pun? treba nam brojati auti na mostu → auti = 0;

→ konstiluo dva reda ujeta : 0 ili 1

→ sujer na mostu = 0

II MONITORI SA LABOSA

wifja popui-se-na-most (sujer) {

mutex_lock (M); \otimes uaci_u-monitor (M)

dok je (auto=3 || (auto>0) && (sujer_na_mostu != sujer))

cond-wait (M, sujer); \otimes { mutex_u_red_wijeta (M, sujer);
uaci_u-monitor (M); }

auto++;

sujer_na_mostu = sujer;

mutex_unlock (M); \otimes izaci_rz_monitora (M)

wifja seti_sa_mosta () {

mutex_lock \otimes

auto--;

also je (auto==0)

sujer_na_mostu! \rightarrow jer nema argumenta!

↑ for (i=0, i<3, i++) {
cond-broadcast (1-sujer); \otimes osloboodi_rz_reda_wijeta (1-sujer);
uaci_u-monitor (M); }

inace

cond-signal (sujer); \otimes osloboodi_rz_reda_wijeta (Xsujer);

mutex_unlock (M); \otimes

III MONITORI SA PREDAVANJA \otimes

D.Z.

① PROGRAMERI

U istoj zgradi nalaze se Microsoftovi i Linux programeri. Zgrada ima samo jedan restoran kojeg programer moraju djeliti. U metu trenutku u restoranu smije biti samo jedna vrsta programera ili je restoran prazan. Svaki programer nema slijedeti odlik:

```
Dreva programer (vrsta) {  
    uti (vrsta);  
    jedi; K.O.  
    izadi ();  
}
```

- a) sinkronizirati programere monitorom
- b) zadatak rješiti bez pojave razgradnjivanja tako da programer jedne vrste nemogu ući u restoran ato na ulazak u restoran N programera druge vrste (kad se jedni najedu drugi jedu)

② PING-PONG

1

U sustavu postaje dvoje vrste dretri. Dretri Ping koje u beskonačnoj petljici ispisuju rječ „ping“ i dretri pong koje u do petljici pišu „pong“. Dretri treba sinkronizirati tako da se ispišuje „ping pong“:

- a) uporabom semafora
- b) uporabom monitora

Rj: 2.

```
a) dreva Ping {  
    dok je(1) {  
        čekaj - BSem1;  
        ispiši („ping“);  
        postavi - BSem2;  
    }  
}
```

```
dreva Pong {  
    dok je(1) {  
        čekaj - BSem2;  
        ispiši („pong“);  
        postavi - BSem1;  
    }  
}
```

POCETNO: BSem1 = 1
BSem2 = 0

b) utička Ping {

```
utic_u_monitor(M); // mutex_lock(M);
dok je (pong == 0) {
    cond_wait(UVrsta.u_red_wijeta(ping, M));
    // utic_u_monitor(M); }
```

ispisi („ping”);
pong = 0;
ping = 1;

oslobodi iz reda_wijeta (pong); //

globalne varijable: ping = 0
pong = 1

dodata Pong {

```
utic_u_monitor(M); // mutex_lock(M);
dok je (ping == 0) {
    cond_wait(UVrsta.u_red_wijeta(pong, M));
    // utic_u_monitor(M); }
```

ispisi („pong”);
pong = 0;

pong = 1;

oslobodi iz reda_wijeta (ping, M); //

cond_signal (ping/pong /M);
mutex_unlock(M);

①

a) utic (vrsta){

```
mutex_lock(M); // utic_u_monitor(M);
dok je (restoran != vrsta) {
    cond_wait(M, vrsta); } // 1)
jedu++;
restoran = vrsta;
mutex_unlock; // zaci_iz_makitora(N);
}
```

globalne varijable: restoran = 0
jedu = 0 // max = 0;

1) uvrstiti_u_red_wijeta (N, vrsta);
utic_u_monitor (M);

3) osloboditi_iz_reda_wijeta (vrsta);

zaci (vrsta) {

```
mutex_lock(M); // utic_u_monitor(M);
jedu--;

```

dok (jedu == 0)

cond_broadcast (1 - vrsta); // 2)

inace

cond_signal (vrsta); // 3)

mutex_unlock (M); // zaci_iz_makitora

}

b) mjeriće sa brojačem koliko ih je poglo → opasnost od tog da od drugih novih N

utic (vrsta) {

mutex_lock (M); // A)

dok je (restoran != vrsta && br > N) {
 cond_wait (M, vrsta); } // 1)

br++;

jedu++;

restoran = vrsta;

mutex_unlock; // B)

}

A) utic_u_monitor (M)

B) zaci_iz_makitora (M)

zaci (vrsta) {

mutex_lock (M); // A)

jedu--;

dok (jedu == 0 || br > N) {

cond_broadcast (1 - vrsta); // 2)

br = 0;

inace

cond_signal (vrsta); // 3)

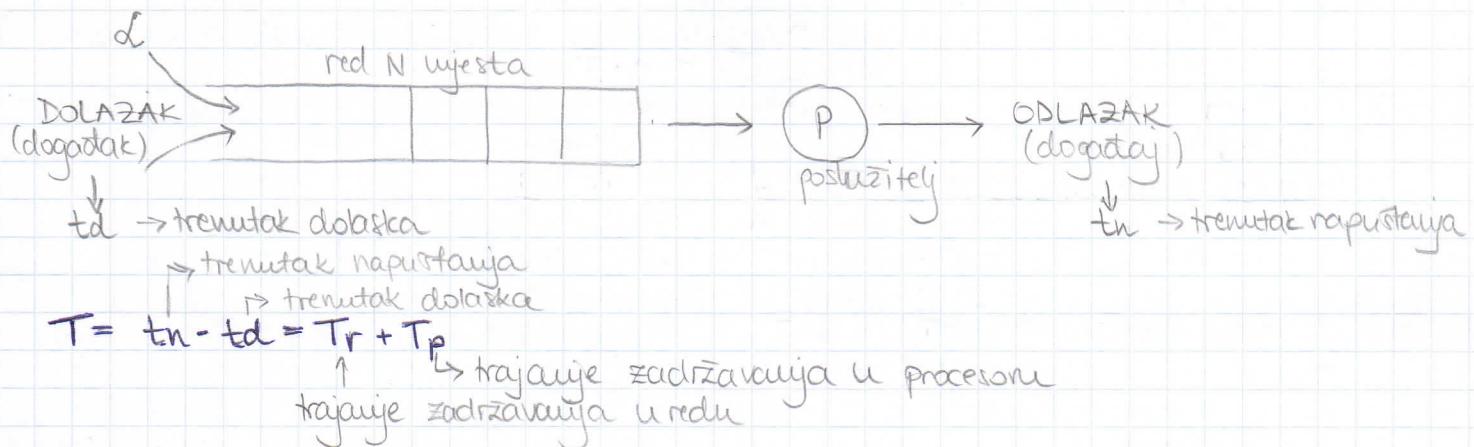
mutex_unlock; // B)

}

2)

for (i=0; i<br; i++) {

F. ANALIZA VREMENSKIH SVOJSTAVA RACUNALNOG SUSTAVA



DETERMINISTIČKO PONAŠANJE \rightarrow sustav gdje znamo trenutke dolaska i napustanja i trajanje zadržavanja u procesu i u redu

npr. punionica lijekova



$$\begin{aligned} T_p &\leq T_d \\ T &= T_p \\ Tr &= 0 \end{aligned}$$

\rightarrow Što procesor radi kada nista ne radi? Vrti dretvu najnižeg prioriteta \rightarrow idle dretva

2 - recipročna vrijednost vremenskog razmaka između dva dolazaka je broj poslova koje u jedinici vremena

3 - recipročna vrijednost trajanja postuzitiranja je broj poslova koje bi postuzitili mogao obaviti u jedinici vremena

f - faktor iskonistaju $f = \frac{1}{\beta}$

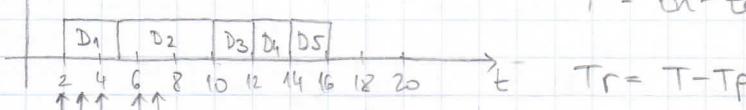
Zad F.1. periodički deterministički sustav
↓ \hookrightarrow M1! poslovi u postuzitiku $T_{uk} = 20$

prijer učijiz:

$$\bar{T} = ? \quad \bar{n} = ?$$

posao	t_d	T_p	b_n	T	Tr
D ₁	2	3	5	3	0
D ₂	3	5	10	7	2
D ₃	4	2	12	8	6
D ₄	6	2	14	8	6
D ₅	7	2	16	9	7

poslovi u procesu



broj poslova u sustavu

$t_h \rightarrow$ trenutak napustanja - citamo sa grafa

$$T = b_n - t_d$$

$$Tr = T - T_p$$

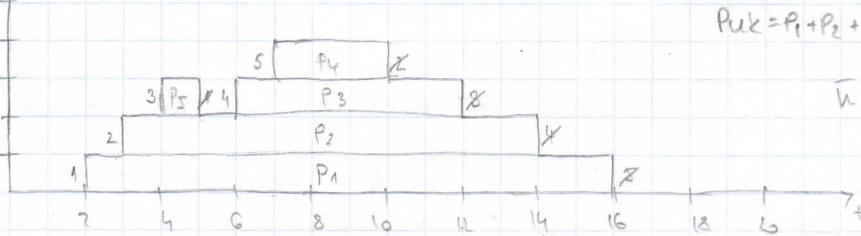
$$\bar{n} = \text{površina ispod grata}$$

$$P_{uk} = P_1 + P_2 + P_3 + P_4 + P_5 = 14 + 11 + 6 + 1 + 3 = 35$$

$$\bar{n} = \frac{35}{20} = 1,75 \quad \bar{f} = \frac{\text{uk. br. poslova}}{T}$$

$$\bar{f} = \frac{5}{20} = 0,25$$

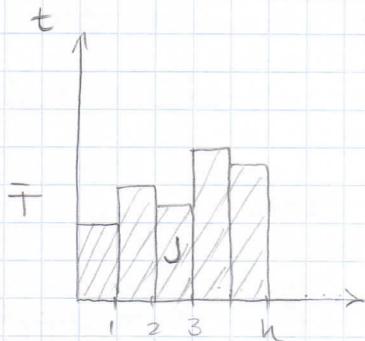
$$\bar{T} = \frac{\bar{n}}{\bar{f}} = 7$$



DOKAZ U ORIGINALU:

$$\text{intenzitet poslova} = \lambda = \frac{n}{T}$$

Zelimo dokazati $\bar{n} = \lambda \bar{T}$



$$J = \sum_{i=1}^n T_i = n \cdot \bar{T}$$

$$\bar{T} = \frac{J}{n} \equiv \bar{n} = \frac{J}{\bar{T}}$$

$$\bar{n} = \frac{J}{\bar{T}} \Rightarrow (J = n \cdot \bar{T}) \Rightarrow \frac{n \cdot \bar{T}}{\bar{T}} = \lambda \cdot \bar{T}$$

NEDETERMINISTIČKI - slučaju dolazi u sustav i slučaju dugo traje
 ↳ način na koji računati \bar{n} i \bar{T}

UVOD U SPROSTSKU

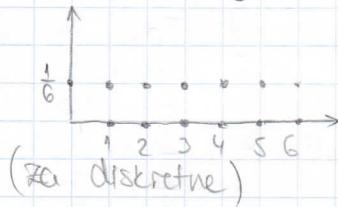
Vjerojatnost dogadaja $\in [0, 1]$

SLUČAJNA VARIJABLA - popravlja pojedinačne vrijednosti s određenom vjerojatnošću

→ DISKRETNAA - popravlja diskrete vrijednosti (kocka $\rightarrow 1/6$ mogućnosti)

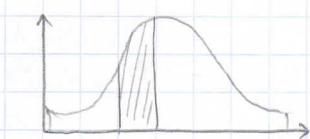
→ KONTINUIRANA - npr. temp.

RASPODJELA VJEROJATNOSTI



na prijenos kocke ona je ista za sve slučajeve

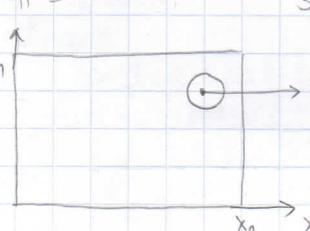
→ prikazuje kako je ukupna vjerojatnost (1) raspodijelena na pojedine vrijednosti slučajne varijable



(za kontinuirane)

vjerojatnost da imamo temp

dokaz



$36,8 \rightarrow 0$

$36,854 - 36,856 \rightarrow mala$

površina točke je 0!

Pitanica :)

$\forall k \in \mathbb{N} \quad \exists (k-1)$ uzastopne prirodne brojeve koji su svi složeni

10^{80} - svi atoni

10^{80-1}

→ Izvodi:
 $p + q$

↑ nije se dogodilo
dogodilo se

QZ: Kolika je vjerojatnost da se u 10 bacanja kocke dobiju 3 šestice?

ima $\binom{10}{3}$ načina za raspodjelu 3 šestice

$$\binom{10}{3} \cdot \left(\frac{1}{6}\right)^3 \cdot \left(\frac{5}{6}\right)^{10-3}$$

↑ 7 puta moramo filati
tri puta moramo pogoditi

$$b(k, n, p) = \binom{n}{k} p^k \cdot q^{n-k}$$

$$b(0, n, p) = q^n = (1-p)^n$$

vjerojatnost da nema dogodjaja

$$b(k > 0, n, p) = 1 - q^n$$

vjerojatnost da ima dogodjaja

POISSONova aproksimacija: $n \rightarrow \infty$, $\lambda = np \rightarrow b(0, n, p) = e^{-\lambda}$

$$b(k, n, p) = ?$$

$$\frac{b(k, n, p)}{b(k-1, n, p)} = \frac{\lambda}{k}$$

$$b(k, n, p) = \frac{\lambda}{k} \cdot b(k-1, n, p) \rightarrow \text{upr. } b(1, n, p) = \frac{e^{-\lambda}}{1!} \cdot \lambda$$
$$b(2, n, p) = \frac{e^{-\lambda}}{2!} \cdot \frac{\lambda^2}{2}$$

$$b(k, n, p) = \frac{\lambda^k}{k!} e^{-\lambda}$$

QZ: U jednoj minuti prosječno padne 100 kapi kiše na površinu stola, koja je vjerojatnost da u sljedećoj sekundi na stol padnu dve kapi?

$$p(k=2, \lambda = \frac{100}{60}) = \frac{\lambda^2 \cdot e^{-\lambda}}{2} = 26,2\%$$

$$p(k, \lambda) = \frac{\lambda^k}{k!} \cdot e^{-\lambda}$$

OČEKIVANJE = SREDNJA NEJEDNOST

$\lambda \rightarrow$ prosječni broj dogodjaja u jedinici vremena

- za proizvoljno vrijeme mijesto jedinicevog: $n \rightarrow nt \rightarrow \lambda t - nt \cdot p \rightarrow \lambda \rightarrow \lambda t$

EKSPONENCIJALNA RAZDIOBA

$$f(u) = \lambda e^{-\lambda u}, \text{ za } u \geq 0$$

$$f(u) = 0, \text{ za } u < 0$$

OČEKIVANJE / SREDNJA VRIJEĐNOST $\rightarrow \frac{1}{\lambda}$

POISSONOVA RAZDIOBA

- vjerojatnost da nema ni jednog dogadaja u intervalu t je $e^{-\lambda t}$

- vjerojatnost da postoji barem jedan dogadaj u intervalu t je $1 - e^{-\lambda t}$

- prosječni broj dogadaja u jedini vrijeme je λ

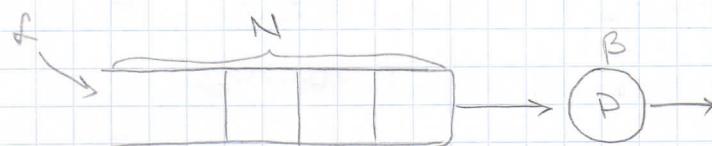
EKSPONENCIJALNA RAZDIOBA

- vjerojatnost da je vrijeme između dva dogadaja veće od t je $e^{-\lambda t}$

- vjerojatnost da je vrijeme između dva dogadaja manje od t je $1 - e^{-\lambda t}$

- prosječno vrijeme između dva dogadaja je $\frac{1}{\lambda}$

PONAVLJANJE



L - broj poslova koji uđu u jedinici vrijeme

$$f - \text{iskorišćivost sustava} = \frac{L}{\beta}$$

$\frac{1}{2}$ - vrijeme između dva posla koja ulaze u sustav

β - broj poslova koje poslužitelj može obaviti u jedinici vrijeme

T - vrijeme odnosno trajanje posluživanja

T - period odnosno ^{trajanje} zadržavanja posla u sustavu

\bar{T} - prosječno trajanje zadržavanja posla u sustavu

n - broj poslova u sustavu

\bar{n} - prosječni broj poslova u sustavu

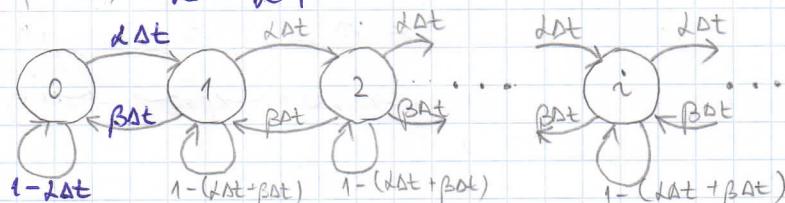
N - kapacitet reda

$N+1$ - kapacitet sustava - $H = N+1$

cilj: izračunati n i f $\rightarrow \bar{n} = \lambda \bar{T}$

- Markovijev lanac

[PM]



- na ulazu: Poisson

- na izlazu: eksponencijala

! vjerojatnosti se NE mogu zbrojati

→ želimo izračunati koliko je p_i

$$p_i(t + \Delta t) = [1 - (\lambda + \beta) \Delta t] p_i(t) + \lambda \Delta t p_{i-1}(t) + \beta \Delta t p_{i+1}(t)$$

ili sam bio u tom
stavju

ili sam prešao
iz stavlja prije

ili sam se
vratio iz stavlja posje

za po drugačije (jer nema prethodnog stavja):

$$p_i(t + \Delta t) = (1 + \lambda \Delta t) p_i(t) + \beta \Delta t p_1(t)$$

→ preko derivacija koje izjednačimo sa o dobijelju sustav (kujiga!)

nakon izvoda dobijemo: $p_i(t) = 1 - \beta \rightarrow p_i = \beta \cdot p^i$

$$p_i(t) = (1 - \beta) \beta^i$$

$$\text{Pa tog slijedi: } \bar{n} = \frac{\beta}{1 - \beta} = \frac{\frac{\lambda}{\beta}}{1 - \frac{\lambda}{\beta}} = \frac{\lambda}{\frac{\beta - \lambda}{\beta}} = \frac{\lambda}{\beta - \lambda}$$

$$\bar{n} = \lambda \cdot \bar{T} \rightarrow \bar{T} = \frac{\bar{n}}{\lambda} = \frac{1 \cdot \frac{\lambda}{\beta - \lambda}}{\lambda} = \frac{1}{\beta - \lambda} = \frac{1}{\lambda} \cdot \frac{\beta}{1 - \beta}$$

Priuđer u kujizi #2.

Zad #2. (POKLON ZAD)

$$\lambda = 2 \text{ s}^{-1}$$

$$N = 5$$

$$\bar{T} = 0,5 \text{ s}$$

$$p(i > N) = ?$$

suprotina vj. = $1 - p(i > N) = p(i \leq N) \rightarrow \text{IZVOD}$ (priuđer iz kuj. #2)

$$\hookrightarrow 1 = \sum_{i=0}^{\infty} p_i(t) \quad \hookrightarrow p(i \leq N) = \sum_{i=0}^N p_i(t)$$

$$p(i > N) = \sum_{i=0}^{\infty} p_i(t) - \sum_{i=0}^N p_i(t) = \sum_{i=N+1}^{\infty} p_i(t) = \sum_{i=N+1}^{\infty} (1 - \beta) \beta^i = (1 - \beta) \beta^N \frac{\beta}{(1 - \beta)}$$

$$p(i > N) = \beta^{N+1}$$

$$\hookrightarrow \beta = \frac{\lambda}{\bar{T}} \quad // \quad \rightarrow \beta = \frac{2}{0,5} = 0,5$$

$$\hookrightarrow \bar{T} = \frac{1}{\beta - \lambda} \rightarrow \beta = \frac{1}{\bar{T}} + \lambda = \frac{1}{0,5} + 2 = 4 \text{ s}^{-1}$$

$$p(i > N) = 0,5^{(5+1)} = 0,5^6 = \frac{1}{64}$$

→ dugi poslovi s parametrima (λ_d , β_d)

→ kratki poslovi s parametrima (λ_k , β_k)

"gejst sustav" :)

$$\lambda = \lambda_k + \lambda_d$$

Koliko troše: $f_k = \frac{\lambda_k}{\beta_k}$

(dugi) $f_d = \frac{\lambda_d}{\beta_d}$

$$\rightarrow f = f_k + f_d$$

$$f = \frac{\lambda}{\beta} \rightarrow \beta = \frac{\lambda}{f} \rightarrow \frac{1}{\beta} = \frac{f}{\lambda} = \frac{f_k + f_d}{\lambda_k + \lambda_d} = \frac{\frac{\lambda_k}{\beta_k} + \frac{\lambda_d}{\beta_d}}{\lambda_k + \lambda_d} = \frac{\frac{\lambda_k \cdot \beta_d + \lambda_d \cdot \beta_k}{\beta_k \beta_d}}{\lambda_k + \lambda_d} =$$
$$= \frac{\lambda_k \cdot \frac{1}{\beta_k} + \lambda_d \cdot \frac{1}{\beta_d}}{\lambda_k + \lambda_d} \quad \left. \begin{array}{l} \text{ao moramo znati} \\ \text{da stavljam KRATKI SU HITRI!} \end{array} \right\}$$

Zad f.3. NE gledati prioritet prema mjestu u redu, nego promijeniti prioritet tako da stavljam KRATKI SU HITRI!

problem: ako pro dođe dugi posao \rightarrow oduzme na procesor

rj: $T_g \rightarrow$ KVANT VREHENJA \rightarrow svaki kvant vremena procesor bi trebao provjeravat dali je dugi proces gotov \rightarrow ako nije u red $T_g = 0.01$ \rightarrow u ovom PR treba naući 2 kvanta vremena za kratki posao

Zad f.4 (?)

Zad. 7.5.

Za neki web sustav s 1 procesorom prosječni broj u min je 100. Dok je snaga poslužitelja znatno veća, on ih može obraditi prosječno 300 u min. Zahtjevi za obradu podliježu Poissonovoj, a mijene obrade eksponencijalnim razloboj. Koliki je postotak poslužiteljskog vremena korišteni za neke druge usluge, a da klijenti i dalje ne čekaju prosječno više od 2s na svjeze zahtjeve.

$$\lambda = \frac{100}{60} = \frac{5}{3} \text{ s}^{-1}$$

$$\beta = \frac{300}{60} = 5 \text{ s}^{-1}$$

$$T' = 2 \text{ s} \quad T = \frac{1}{\beta - \lambda} = \frac{1}{5 - \frac{5}{3}} = 0,3 \text{ s}$$

$$\lambda' = \lambda \quad \beta' = ? \quad \beta = \frac{1}{T} + \lambda = \frac{1}{2} + \frac{5}{3} = \frac{13}{6} = 2,16 \text{ s}^{-1}$$

$$\frac{\beta'}{\beta} = 43,3\% \leftarrow \text{radi na webu}$$

$$1 - 0,43 = 56,6\% \text{ rezervirati za druge poslove}$$

Zad 7.6

Za neki web sustav s 1 poslužiteljem prosječni broj zahtjeva u sec je 100.

Njegovi: Poisson na ulazu, eksponencijala na izlazu. Poslužitelj obrađuje 3 tipa zahtjeva z1, z2, z3. Za zahtjeve tipa z1 obrada prosječno traje 5ms, z2 8ms, a z3 10ms. Ukoliko je postotak zahtjeva z1 → 30%, z2 → 40%, z3 → 30% odrediti prosječnu kvalitetu usluge (prosječno vrijeme zadizavanja u sustavu T) i verovatnošću da se u sustavu nalazi više od 10 zahtjeva?

$$\lambda = 100 \text{ s}^{-1}$$

$$T = ?$$

$$\lambda_1 = 30 \text{ s}^{-1}$$

$$P(i > N) = ?$$

$$\lambda_2 = 40 \text{ s}^{-1}$$

$$N=10$$

$$\lambda_3 = 30 \text{ s}^{-1}$$

$$1 = 5 \cdot 10^{-3} \text{ s} \rightarrow \beta_1 = 200$$

$$\beta_1 = \lambda_1 = 0,15$$

$$1 = 2 \cdot 10^{-3} \text{ s} \rightarrow \beta_2 = 125$$

$$\beta_2 = \lambda_2 = 0,32$$

$$1 = 10 \cdot 10^{-3} \text{ s} \rightarrow \beta_3 = 100$$

$$\beta_3 = \lambda_3 = 0,3$$

$$\beta_{uk} = \beta_1 + \beta_2 + \beta_3 = 0,15 + 0,32 + 0,3 = 0,77$$

$$\beta = \frac{\lambda}{T} = \frac{100}{0,77} = 129,87$$

$$\bar{T} = \frac{1}{\beta - \lambda} = 0,033 \text{ s}$$

$$P(i > 10) = \beta^{10} = 0,77^{10} = 0,0564$$

Zad F.F.

U nekom poslužiteljskom centru naciđeno je analiza rada poslužitelja.

Poslužitelj P_1 prosječno dobiva 70 zahtjeva u min i njegova prosječna iskoristivost je 20%. P_2 prosječno dobiva 200 zahtjeva u min i njegova prosječna iskoristivost je 30%. $P_3 \rightarrow 150$ zahtjeva u min $\rightarrow 10\%$. P_3 je processki najjači (50% je jaci od P_1 i dvostruko od P_2). (Poisson na ulazu, exp na izlazu). Izračunati kvalitet usluge (T) kada bi se svi poslovni preseleli na poslužitelj P_3 .

$$\delta_1 = \frac{70}{60} = 1,16 \rightarrow 20\% \quad \beta_1 = 0,2$$

$$\delta_2 = \frac{200}{60} = 3,3 \rightarrow 30\% \quad \beta_2 = 0,3$$

$$\delta_3 = \frac{150}{60} = 2,5 \rightarrow 10\% \quad \beta_3 = 0,1$$

kada se posao sa 1 preseli na 3 $\beta_{31} = 1,5\beta_1$
 $\beta_{32} = 2\beta_2$

$$\delta = \delta_1 + \delta_2 + \delta_3 = 6,9$$

$$\beta = \beta_1 + \beta_2 + \beta_3 = 0,2 + 0,3 + 0,1 = 0,6$$

$$\delta = \beta_1 + \beta_{32} + \beta_3 = \frac{\delta_1}{1,5\beta_1} + \frac{\delta_2}{2\beta_2} + \delta_3 = \frac{0,2}{1,5} + \frac{0,3}{2} + 0,1 = 0,383$$

$$\beta = \frac{\delta}{\delta} = \frac{6,9}{0,383} = 18,26$$

$$T = \frac{1}{\beta - 1} = \frac{1}{18,26 - 6,9} = 0,08$$

8. GOSPODARENJE SPREMNICKIM PROSTOROM (3. CIKLUS)

IZVORI ADRESA:

→ adresa u programskom logiku

→ stupanj

→ unutar instrukcija

- svaki proces djeluje u svom adresnom prostoru i ne može van gledati

↳ to je dobro za vježbenički rad

↳ nove loze za višezadaci rad → zato su se zaustavili direkte koje mogu sve isto

- unutar svakog direktnog podprostora može se propisati:

- instručni direktive

- stupanj

- podatkovni

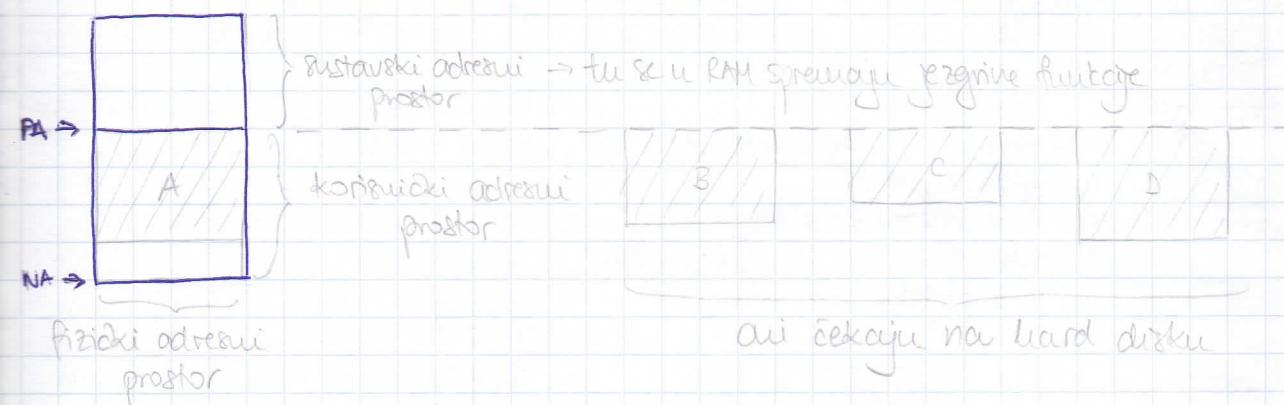
- zajednički spremnički prostor procesa sadrži:

→ zajedničke podatke

- "prazni dio" - hrpu spremničkih lokacija odake direkte mogu dinamički dostavljati potrebne spremničke lokacije i kada se vradiju isplažuju spremničke lokacije

→ mogu li se na takvom radnom prostoru smjesti procesi čiji adresni prostor prelazi
tu veličinu?

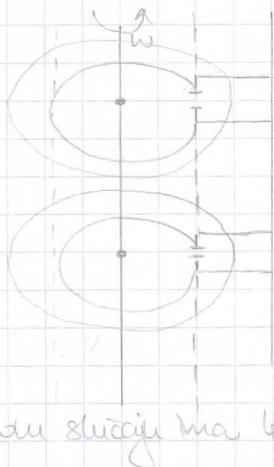
↳ imamo 64 bitna racionala \rightarrow programi do 2^{64} veličine



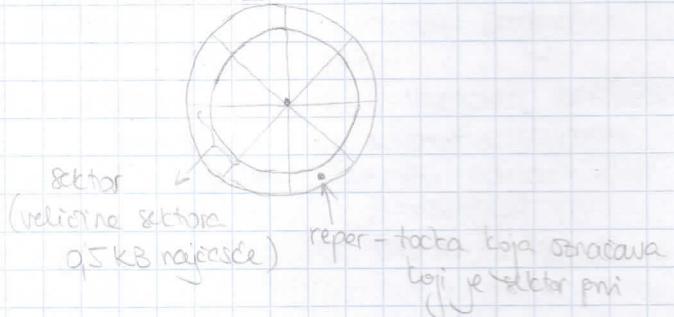
MAGNETSKI DISKOVI (danas su poznati spremnici) \rightarrow preko DMA se spaja na procesor

- imaju drugačije uloge:

→ kao pomoći ili dopunski



(pogled od ozgona)



u okviru slučaju ima 4 staze(?)

→ ne može se citati sve odjednom → samo 1 glava cita

→ kada cita/pise sektor → sektor se puno pomeravlja u međuspremnik preko satirnice u RAM

VELIČINA: od 1 do 5,25 inca → 2,5 i 3,5 (najčešće)

BRZINA STAZE: 250 000 staze po inču

EUSTOČA BITOVA: 10 - 500 GB po inču

BR. OKRETJA U MIN: 5400 - 1500 → 7200; 10 000 (najčešće)

DISKOVNA JEDINKA: 1-20 plota

BR. STAZE NA DISKU: 2000 - 400 000

STAZE INA: 300 - 30 000 sektora

KAPACITET SE MOže u terabajtima (od 2007)

trajanje prijevoza podataka

→ ukupno trajanje prijevoza podataka. (PITANJE M1)

↳ trajanje postavljanja glave (head position time)

↳ trajanje traženja staze (seek time):

1. vibriranje
2. konst. brzina
3. usporavanje
4. fini podešavanje

↳ rotacijsko korištenje (rotation)) = TR/2

↳ trajanje prijevoza podataka (data transfer time)

↳ trajanje citanja staze ili delu staze

! treba uzeti u obzir faktor preplitanja

↳ trajanje premještaja glave sa staze na stazu

(treba uzeti sva 4 koraka u obzir jer su staze dale)

$$n | n+1 | n+2 | \dots | \dots | n+k-1 | n+k | n+k+1$$

Def: Ako su 2 sektora koji logički dijede jedan rezultat, fizicki vrijestvi na sektoriima N i N+k-1 tada je faktor preplitanja (interleave factor) jednak k

? standardno pitanje → w = 7200 okr/min → TR = ? (D.Z.)

$$w = \frac{7200}{60} \text{ okr/s}$$

$$TR = 8,33 \text{ (trajanje 1 okretaja)}$$

Zad 8.1.

Disk ima 128 sektora po stazi veličine sektora 512 byteova, 2000 sektora i vrati se brzinom 4800 okr/min. Upravljanje sektora potreba jedan cijeli sektor u intervu spremnik a zatim ga prenosi u globalni spremnik. Prijenos u WS odvija se brzinom 10 Mbit/sec, a za to vrijeme sektop ne može otidati s diska.

a) Koliki treba biti faktor preplitanja?

$$\omega = 4800 \text{ okr/min} = 80 \text{ okr/s} \rightarrow T_R = \frac{1}{\omega} = 0,0125 \text{ s} \quad (\text{trajanje jednog okretaja})$$

$$T_R = ?$$

$$t_S = \frac{T_R}{128} = 9,76 \cdot 10^{-5} \text{ s} \quad (\text{čitajuće jednog sektora})$$

$$t_{SW} = \frac{S}{t_S} = \frac{S}{\frac{1}{128}} = 128 \cdot S \quad (\text{trajanje prijenosa 1 sektora u gl. spremnik})$$

$$t_{SW} = \frac{\text{broj byteova}}{\text{brzina}} = \frac{512 \cdot 8}{10 \cdot 10^6} = 4,096 \cdot 10^{-4} \text{ s}$$

$$\frac{t_{SW}}{t_S} = 4,194 \quad (\text{priješeni sektori}) \rightarrow k = 6$$

(polog sas u mali fulali, ali sas fulali)

b) Koliko prosječno traje prebacivanje KOMPACTNOG snijesterne datoteke veličine 235 KB ako je vrijeme postavljanja (trek) jednako 12 ms, a vrijeme prenještavanja sa staze na stazu 20 ms.

B - byte

b - bit

K - 1024

k - 10^3

$$64 \text{ KB je jedna staza} \rightarrow \text{br. staza} = \frac{235}{64} = 3,67$$

$$t_S = k \cdot t_S \quad (\text{prosječno čitajuće jednog sektora})$$

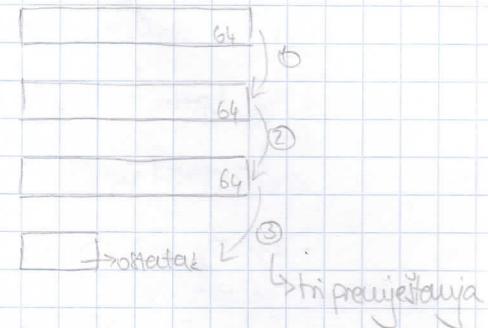
$$\textcircled{1} \quad t_{PG} = t_{trek} + \frac{T_R}{2} \quad (\text{trazišmo početak datoteke})$$

$$\textcircled{2} \quad 235 \cdot 2 \cdot \frac{\text{KB}}{\text{sektor}} \quad (\text{trajanje čitanja sektora})$$

$$\textcircled{3} \quad 3 \cdot \left(\frac{t_{prem} + T_R}{2} \right) \quad (\text{prenještavanje sa staze na stazu})$$

\uparrow broj-prenještavanja

$$\textcircled{1} + \textcircled{2} + \textcircled{3} = 318,4 \text{ ms} \quad (\text{kao da je})$$



\rightarrow moguć rezultat kao iz kuj.

Zad 8.2.

Disk imao 100 sektora po stazi veličine 1KB, 2000 staza, 4 plote i vrati se brzinom 200 otv/min. Podaci su zapisani na obje strane plote. Upoznajteći sklop prošta jednu cijelu stazu u interni spremnik, a zatim prenosi interne podatke u glavni spremnik (tekuće rade moderni diskovi). Prijenos u glavni spremnik odvija se brzinom 20MB/s i za to vrijeme sklop ne može čitati s diska.

- D.2.
- Koliki je kapacitet diska?
 - Koliko će prosjekom trajuće prebacivanje kompaktnog svijetla datoteke velicine 135KB, ako je vrijeme pozitivanja ~10ms i vrijeme preplitanja svih staza na stazu ~1ms.
(\Rightarrow KOMPATNO! - datoteka je tako moguće bude sklopana)
 - Koliko iznosi faktor preplitanja ako disk vrati sektor po sektor, te koliko traje prijenos podataka u tom slučaju?
- (svaki put u zadacima k je 1024)

b) uputa:

$$1) \text{ trajanje tražnja staze} \Rightarrow 10\text{ms} \quad] \text{ trajanje tražnja početka datoteke} \\ \text{stacijensko korištenje} \Rightarrow \frac{\text{trajanje}}{2} \quad] \quad 2 \rightarrow 4,17\text{ms}$$

$$2) \text{ Čitaće cikle staze} \Rightarrow T_c = 8,33 \quad (\text{za 1 okretaj pročitavamo cijelu stazu}) \\ \text{O da marni faktor preplitanja onda je } A \cdot T_c$$

$$3) \text{ trajanje prijenosa podataka (look)} = \frac{\text{br. bitova}}{\text{brzina}} = \frac{100 \cdot 1024 \cdot 8}{2 \cdot 10^7} = 40,96 \text{ ms}$$

$$3b) t_{prem} = 1\text{ms} \quad (\text{memori se uzmatti u obzir ako datoteka stane na 1 cilindar}) \\ \rightarrow \max(3a, 3b) \rightarrow = 40,96\text{ms}$$

$$4) \frac{T_c}{2} = \frac{4,17\text{ms}}{2}$$

$$5) = 2) \frac{3,33\text{ms}}{2}$$

$$6) \text{ prijenos podataka } 35\text{KB} \quad \frac{35 \cdot 1024 \cdot 8}{2 \cdot 10^7} = 14,34\text{ms}$$

$$\Rightarrow 10 + 8,33 + 40,96 + 4,17 + 3,33 + 14,34 = 80,3\text{ms}$$

$$D.Z. Q = 173,59\text{ms}$$

- U tablicama operacijskog sustava za svaki se proces mora nalaziti popis sektora diska u kojima se nalazi suvjeteni program.
- Nadi na logu su sektori suvjeteni na disku mogu jato utjecati na trajanje prebacivanja programa

PR 8.2. iz knjige str. 197 → može pokazati koliko je vremena da datoteke ne budu fragmentirane

procesui informacijski blok ~ opisnik oblike (procesa)

↳ pokrivaaju podatke o procesu

↳ sadrži se: - sve ostale info. važne za odnajdijanje procesa

- sve informacije povezane s gospodarskim spremnicim prostora
- tablice sektora u kojima je program procesa pohranjen na disku
(tućemo naći i podatke o suvjetovanju programa u sadržaju spremnika)

→ STATIČKO RASPOREĐIVANJE RADNOG SPREMNika

- apsolutne adrese } slika 8.6. str 206
- relativne adrese

- programi se uisu mogu jednostavno preseliti jer to selektuje zaliđenja mera generirajuće apsolutnu adresu programa

podjela spremnika na partice → slika 8.7. str. 207

tijekom rada spremnik neće biti potpuno iskoristiven zbog dva razloga:

1) unutarnja fragmentacija

(RAM) ↳ jer su programi manji od prostora, pa se njihova mračna u prostoru, ta mračna je ekvivalentna → fragmentacija
(disk) ↳ na disku će ta mračna (odnosno unutarnja fragmentacija) javljati kada spremniku datoteku i ostane naručiti ostatak

2) vanjska fragmentacija

pričkom statickog raspoređivanja
↳ jer nemačke particije RAM-u stoje prazne mreže suvjetne preplne

→ DINAMIČKO RASPOREĐIVANJE RADNOG SPREMNika → slika 8.8. str. 203.

BLIC → gdje se generiraju adrese: PC, kroz koja sloga, adrese unutar instrukcija

MMU → memory management unit

→ fragmentacija kod dinamičkog raspoređivanja → poređenje sa programom po redu
kada neki program završi stvorit će RUPA → slika 8.11 str. 205.

PROCESNI ADRESNI PROSTOR ↲ segment u koji se pokrivaju instrukcije
→ slobodni segment
→ podatkovni segment

suzbijajuće fragmentacije:

- pri novom zaliđenju potrazi se najmanja RUPA u kojoj se mogu suvjetiti novi program
- pri svakom oslobadanju (ostanak OS!) usvajastala mračna spaja se s eventualnim susjedima mračna u tomu vrednu mračnu

PR 8.3. → str. 206.

IZVOD → Knutheov SOV algoritam

$$p+g=1 \quad p \gg g \quad p \approx 1 \quad g \approx 0$$



ostobodimo



vjerojatnost da se broj nupa poveća za 1 = broj da se smenji za 1

vjerojatnost ostobodavanja

puni brokova imenik $m = A + B + C + D$

$$p^{\uparrow} - p^{\downarrow}$$

$$p_0 \frac{a}{m} = p_0 \frac{d}{m} + p_2 \cdot g$$

\downarrow vjerojatnost da se smenjuje
da se smenjuje
ostobodio a ostobodio d
(tj. da se ne
ostobodilo)

vjerojatnost da
zadnje su ob u tri

p_0 - vjerojatnost ostobodavanje.

p_2 - vjerojatnost zavreća

m - puni brokovi

n - broj nupa

$$\frac{q}{m} = \frac{d}{m} + (1-p) \rightarrow q = dt(1-p)m$$

$$m = d + (1-p)m + b + c + d \rightarrow [B=C]$$

$$m = 2d + m - pm + 2b$$

$$m = \frac{b+d}{2} + d = \frac{2b}{2} + d = b + d$$

$$pm = 2m \rightarrow \boxed{m = \frac{m}{2}}$$

PREKLOPNI NAČIN UPORABE RADNOG SPREMINIKA → puni u ponijesti

↓ DOBJEĆIVANJE SPREMINIKA STRANCIENJEM (od stranica) → koristi se danas

slika 8.14. str. 21

→ PRIMJERI za domaća (8.4; 8.5)

Faz 8.3.

U sustavu sa stranicama programne veličine 400 ječi (1 - 400) generira sljed adresa : 23, 47, 333, 81, 105, 1, 400, 157, 30, 209, 149, 360

Program ima na raspolaganju 200 ječi radnog spramnika. 1) Napisati

už referenciranju stranica veličine 50 ječi! 2) Koliko je početek promatranja stranica

za sve tri strategije izbacujući? 3) Prikazati trenutni izgled tablice prenosenja

programa na kiju primijene optimalnu strategiju!

1)	23	47	333	81	105	1	400	157	30	209	149	360	8 stranica i 40kura
stranice	1	1	7	2	3	1	8	4	1	5	3	8	

FIFO	-	1	1	1	1	1	8	8	8	3	3	3	spremnik se puni diagonalno
-	-	7	7	7	7	7	4	4	4	4	4	8	
-	-	2	2	2	2	2	2	1	1	1	1	1	
-	-	3	3	3	3	3	3	3	5	5	5	5	
LRU	-	1	1	1	1	1	1	1	1	1	1	1	
-	-	7	7	7	7	7	8	8	8	3	3	3	
-	-	2	2	2	2	2	4	4	4	4	8	8	
-	-	3	3	3	3	3	3	3	5	5	5	5	
OPT	-	1	1	1	1	1	1	1	1	5	5	5	
-	-	7	7	7	7	7	8	8	8	8	8	8	
-	-	2	2	2	2	2	2	4	4	4	4	8	
-	-	3	3	3	3	3	3	3	3	3	3	3	
LFU (+LRU)	-	1	1	1	1	1	1	1	1	1	1	1	
-	-	7	7	7	7	7	8	8	8	3	3	3	
-	-	2	2	2	2	2	2	4	4	4	8	8	
-	-	3	3	3	3	3	3	3	5	5	5	5	

↑ ↓ ← →
Pocetak promatranja Pocetak

2) imamo 12 zadjeva

Za FIFO \rightarrow 10 promatranja $\rightarrow \frac{10}{12} \cdot 100\%$

Za LRU \rightarrow 9 promatranja $\rightarrow \frac{9}{12} \cdot 100\%$

Za OPT \rightarrow * $\frac{7}{12} \cdot 100\%$

Za LFU

3) Ibroj stranica bit prisutnosti

1	1	1
2	3	0
3	2	1
4	3	0
5	4	1
6	-	0
7	2	0
8	3	1

FIFO - first in, first out

OPT - gleda u budućnost
(ako će ti trebati sre, bacamo onu koja će vam ujedalje u budućnosti konisti)

LRU - last recently used
izbacuje se ona koja

LFU - last frequently used
(najmanje u prošlosti konisti)

→ jedan gleda koliko puta
se javlja određeni broj (LFU?)
pa onda od svih koji su te
istu broj puta ponavljali
konisti LRU

ACCES BIT

BIT OISNOCE - nečistu stranicu treba znati pohraniti na disk, pa se onda otkrije
koji dijelovi nisu uključeni i moguće je ih ukloniti

treba stranice treba podjeliti na 4 stranice:

tip	A	B	
0	0	0	cista i nisu uključeni → idealno za pohraniti
1	0	1	nečista i nisu uključeni → pohraniti učinak je nečista
2	1	0	cista i uključena nisu uključene → ako nema nihog, onda ona
3	1	1	nečista i uključena nisu uključene → nagon sluci

Tako se ipak ne konisti jer je prekomplikiran → svakore išveri se margini spustiti na hardware

SATNI MEGHANIZAM / SATNI ALGORITAM

↳ kružni ciklus radica uve stratezije

FIFO strategija → dobitna kompl. za više procesorsku radnalu

→ os se odnosi na jednoprocesorsku radnalu
lista organizirana kao FIFO ali zadnji pokazuje na prvi → kreću lista stranica
konisti se jedna kataljka

kada se radi str. kde se treba izbaciti i gde se dolje dati pristupa (A) - 1 prethodi
radi 1 u 0 i ide se dalje, ona koja ima 0 u nju se briše, onda se dodaju novi i
postavlja se A=1

važnije → da se sve razlikuju (tj. od pre do zadnje i sve sa 1 staninom 0)

(prstena: kružni radnički)

21: → pitanje opisati satni mehanizam

u skraćenosti: manovih procesa nisu dreti → KAKO GOSPODARI OKVIRIMA?

? gdje se nalazi procesni informacijski blok → u sustavskom dijelu spremnika

okvir → učitajnu LKB → pripada 1 procesu
djeleći okvir → oni su u zajedničkom spremniku

stanja okvira: aktivni okvir → dodjeljeni procesu i koniste se
slobodni → pohranimo ih u listu slobodnih točkutno se ne koniste
↳ kod stanjima u listu treba potpisati

? što će ih potpisati? hardware na inicijatoru A-a
potrebna direkta (koja je uska) i brise okvir i starta u listu sa obrisanim...
obrisati tek moramo dodjeliti

medusnica

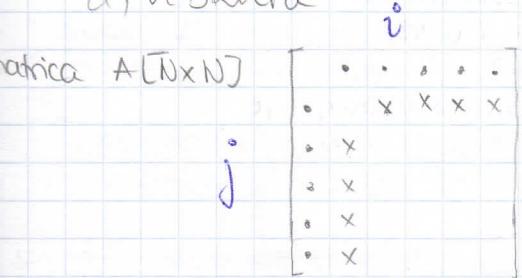
↳ lista neispravnih okvira (space shuttle)

dodata okvire procesima

Zad 8.4.

U sustavu s virtualnim spreminjatorima veličina okvira je n i jedi i okvir se puni na zaliđev (stranica je u zaliđev). Algoritam za mijenjanje stranica je LRU. Poredak: $A[1..N \times 1..N]$ je pohranjen po redima (na susjednim lokacijama se mijenja desni index). Koliko promatralja će razvati prikazani program ato za poredak A u rednom spreminjatoru postoji?

- a) 1 okvir
- b) 2 okvira
- c) 3 okvira
- d) n okvira



redci koji se mijenju pohraniti:

$$a) \quad \begin{matrix} 1 & 2 & 1 & 3 & 1 & 4 & \dots & 1 & N & 1 & 2 & 3 & 2 & 4 & \dots & 2 & N & \dots & N & N-3 & N-2 & N-3 & N-1 & N-3 & N & N-2 & N-1 & N-2 & N & N-1 & N \end{matrix}$$

lansova: $2(N-1)$

$$b) \quad \begin{matrix} 1 & 1 & | & 1 & 1 & | & 1 & 2 & 2 & | & 2 & | & 2 \\ -2 & 3 & | & 4 & | & N & N & 3 & | & 4 & | & N \end{matrix}$$

promatralja: N

$$t = \emptyset$$

Za $i=1$ do $N-1$:

$$\begin{aligned} \text{za } j=i+1 \text{ do } N \{ \\ t = t + A[i, j]; \\ t = t + A[j, j], \end{aligned}$$

bitno name je
čemu se pristupa

$$\begin{matrix} i=1 & j=2 \dots N \\ i=2 & j=3 \dots N \\ i=N-1 & j=N \end{matrix}$$

$$i=N-3 \quad i=N-2 \quad i=N-2 \quad i=N-1$$

(A)

(B)

(C)

$$c) \quad \begin{matrix} 1 & 1 & | & 1 & 1 & | & 1 & 3 & 3 & | & N-1 & N-4 & N-4 & N-2 & | & N-2 & N & N & | & | & | \\ -2 & 2 & | & 4 & | & N-1 & 2 & 2 & | & 2 & | & N-1 & N-3 & N-3 & N-3 & | & N-3 & N-3 & N-1 & | & | & | \\ 3 & 3 & | & N & N & | & N & 4 & | & N & | & N & N & N & N-1 & | & N-1 & N-2 & N-2 & | & | & | \end{matrix}$$

$$d) \quad n \text{ okvira} \rightarrow n \text{ promatralja}$$

? Rastojanje odje $\times 2$?

$$\text{(A)} \quad 2 \sum_{i=1}^{N-1} i = 2 \cdot \frac{(N-1)N}{2} \rightarrow \sum_{i=1}^N i = \frac{N(N+1)}{2}$$

$$\text{(B)} \quad \sum_{i=1}^N i - 2 = \frac{N(N+1)}{2} - 2$$

$$\text{(C)} \quad \sum_{i=1}^N i - 4 = \frac{N(N+1)}{2} - 4$$

korijeno LRU, FIFO, OPT
nove stranice idu u rednu redoslijedu

7

PR. 8.13

$\rightarrow 30S$

$\rightarrow 8h \rightarrow$ ne postavlja novin gospodarenje spreminjatoru

Zad 8.5:

U nekom sustavu trebaju se obaviti 4 programa: P_1, P_2, P_3, P_4 koji su red apprenuljeni u zadataku spominu N. Zauzimaju redom 5, 8, 3, 10 MB. Pogodjaj potrebujući i završetke programa poznati su u naredjed.

- $t_1 \rightarrow P_1$ pokreće (5MB)
- $t_2 \rightarrow P_2$ pokreće (8MB)
- $t_3 \rightarrow P_1$ gotov (5MB free)
- $t_4 \rightarrow P_3$ pokreće (3MB)
- $t_5 \rightarrow P_3$ gotov (3MB free)
- $t_6 \rightarrow P_4$ pokreće (10MB)
- $t_7 \rightarrow P_4$ završava (8MB free)
- $t_8 \rightarrow P_4$ završava (10MB free)

Sustav ima na raspolaganju 20MB RAM-a.

Pričazati stanje RAM-a kada se koriste metode upravljanja:

- staticko upravljanje s relacijom od 10MB
- dinamicko upravljanje
- stranicanje, uz relaciju od 1MB

a) b
→ 2xvela segmenta: $\frac{20}{10} = 2$

t_1	P_1	P_2
t_2	P_1	P_2
t_3		P_2
t_4	P_3	P_2
t_5		P_2
t_6	P_4	P_2
t_7	P_4	P_2
t_8		

b)

t_1	P_1	P_2
t_2	P_1	
t_3		P_2
t_4	P_3	P_2
t_5		P_2
t_6		

P_4 NE STANE NITI U JEDNU RUPU! → mora dekati da P_2 zavri

c)

t_1	P_1		P_2		P_3		P_4
t_2	P_1		P_2				
t_3			P_2				
t_4			P_2				
t_5			P_2				
t_6			P_2				
	- - - - -		P_2				P_4
			P_2				P_4
	P_4		P_2		P_3		P_4
t_7							
t_8							

ZAD OD 8.1 DO 8.5 → MI!

Ponavljajuće:

PITANJA IZ UDŽBENIKA ZA MI:

- poslužiti registar projekti služi za ostvaruje inačice LRU strategije (gleda vremenski dio a ne cijeli projekt)
- bit očitocé: može biti str. mijenjana ili nije nalazi se u tačici strojnicu u tačici prevođenja (među tih 12 bita)
- tačica prevođenja služi za prevođenje logičke u fizicku adresu
zadnjih dva bita izravnju prelazi (\rightarrow to je adresa unutar stranice) \rightarrow 12
- satni algoritam je inačica LRU (opisati ga)
- pojedini okvir se može nalaziti u stanju: aktivni okvir, slobodi okvir, slobodni s (ne)potrebljivim sadržajima
- TLB je međuspremnik koji se nalazi u procesoru u MMU (često MMU - memory management unit)
- fizicki add prostora o RAM-u
logički add prostor ovisi o arhitekturi računala

9. DATOTEČNI PODSUSTAV

DATOTEČNA KAZALIKA - kazalika koja pokazuje na adresu u sustavskom dijelu radnog spremnika gdje se nalaze svii podaci o datoteci uključujući opisnik datoteke

pojedna prema fizičkom datoteka: **STRUKTURIRANE**
NESTRUKTURIRANE

- datoteke se sastoje od zapisa, svaki zapis imao svoja svojstva (to se koristi kod baza)

! svaka datoteka imao svoj opisnik: Inačiv datoteka

- tip datoteke
- lozinka
- ime vlasnika datoteke
- prava pristupa
- vrijeme stvaranja datoteke
- vrijeme ~~započinje~~ uporabe datoteke
- ime posljednjeg korišćenika i vrijeme
- opis ~~sadržaja~~ datoteke na disku

Kako se datoteke snimaju na disk:

- najmanja jedinica datoteke je sektor, može biti 2ⁿ sektora \rightarrow klaster (2K, 4K, 8K itd.)
- disk se može podjeliti na particije i nazvati ih kako god
 - \rightarrow svaka particija imao ~~datotemu~~ tačicu sa opisnicima

Kako videti evidenciju o slobodnom prostoru na disku:

\rightarrow (najjednostavniji): bitovni prikaz

- | \rightarrow lista sa sektorima i bitom sauzete (1 - sauzete)
 - kod losih sektora on formatoraju stavka 1
- \rightarrow inačica bitovnog prikaza sa listom (slika 9.6 na str 246)

Kako smjestiti datoteke na disk?

- najjednostavnije postaci kada su datotece smještene na uzastopne sektore

fragmentacija - nemogućnost koristeći cijeli prostor diska

→ nedostaci u obliku stavljanju jedne kraj druge

→ npr: odisk nije fragmentiran → datoteka je fragmentirana

PR 9.5 i PR 9.6 → rješavati će se u sni zadatku!

npr. 1024B sektor
32 bitna arhitektura } 256 kazaljki

LINUX / UNIX

10 pravim direktnih kazaljki = do 10K



- INDIREKTNA KAZALJKA koja pokazuje na sektor ~~preko~~ kazaljki
- ukupna veličina 11 kazaljki je 256K \$ 256
- prvostrukos INDIREKTNA KAZALJKA pokazuje na sektor ~~prethodnog~~ ali te kazaljke nisu na datoteku nego na 256 sektora.
- kapacitet: $10K + 256K + 256 \cdot 256K =$ datoteče do 64M
- drugosstrukos INDIREKTNA KAZALJKA pokazuje na sektor s 256 kazaljki koje pokazuju na sektore s 256 kazaljki a svaka kazaljka pokazuje na 256 sektora
- kapacitet: do 40GB datoteke

WINDOWS NT

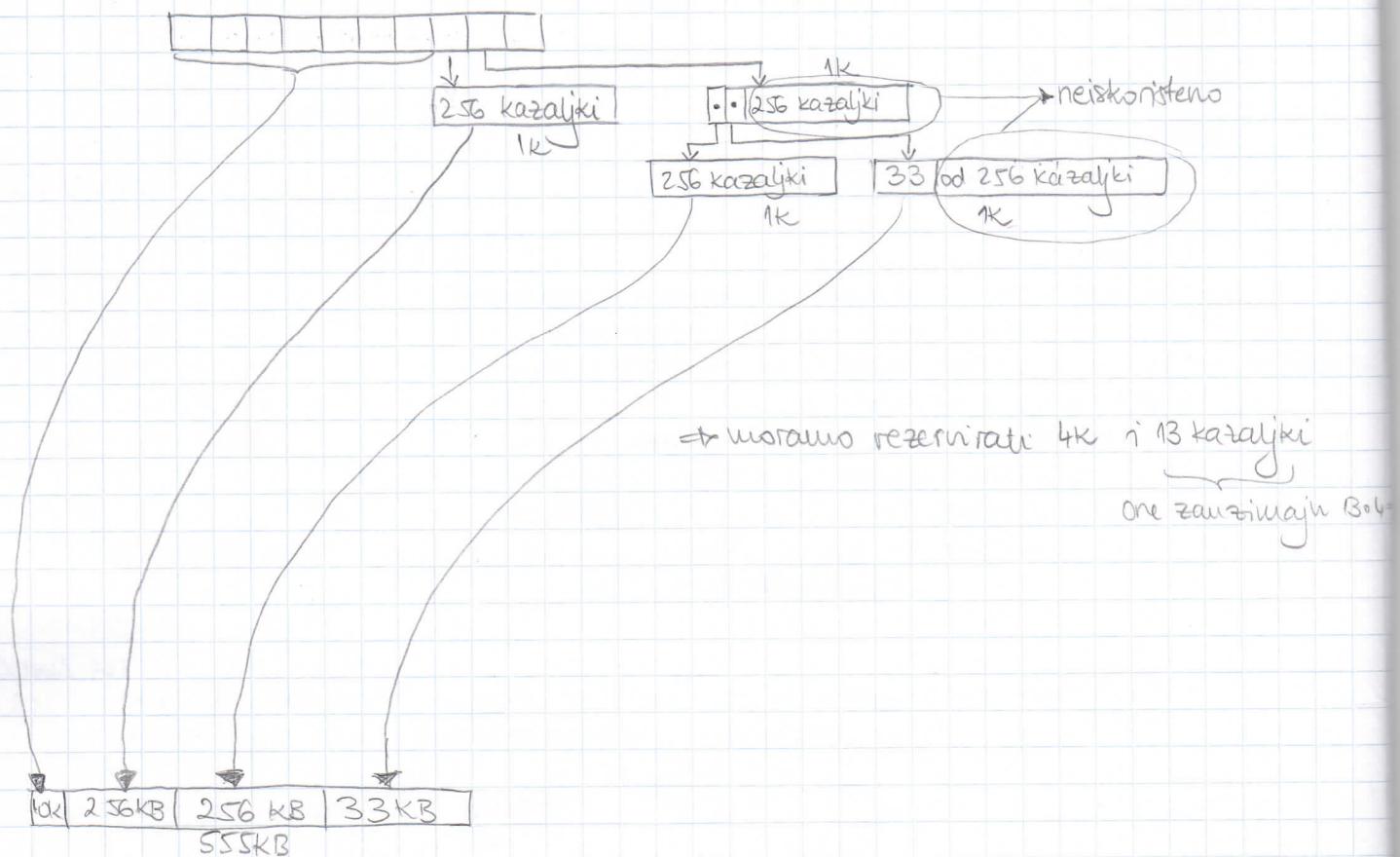
LCN - redni broj klostera = kolika je najveća datoteka može biti

MASTER FILE TABLE (MFT) = glavna tablica datoteka

- male datoteke su pohranjene u MFT!

9.2. Na UNIX datotečnom sustavu pohranjena je datoteka veličine 555KB. Koliko se prekucickog prostora zauzimaju kazaljke za tu datoteku u tablici dodatčnog sustava skicirati organizaciju tih kazaljki! Velicina sektora 1024 okreta (1K), velicina kazaljka 32 bita (4B).

555KB
sektor 1KB → 1024
32 bita



Koja je najveća moguća datoteka koja se može pohraniti u ovakvom sustavu?

Zad 9.3: Na slici su prikazane nakupine sektora (klasteri) nekog vrednog (particije). Nakupine sektora jedne datoteke su zasjenjene i numerisane redoslijedom kako se logički pojavljuju u datoteci (masno otisnute brojke). Opisati datotečnu tačicu, tj. ujem dio koji odstoji najčešće zadane datoteke. Redni brojni nakupine sektora način davanje prikazani su u gorenjem lijevu kutu svake zelje. Velicina nakupine sektora za ovaj primjer nije važna i može biti priuđenice 4K. (Za NTFS, UNIX i FAT)

↳ ne treba zauzeti

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

NTFS:

VCN	LCN	br. naktynia
1	37	2
3	18	3
6	57	5
11	51	5

} audito redaka koliko iuva naktynia

UNIX:

37	38	18	19	20	57	58	59	60	61	X
----	----	----	----	----	----	----	----	----	----	---

↓

51	52	53	54	55
----	----	----	----	----

ostatak kazaliki nieskorysten