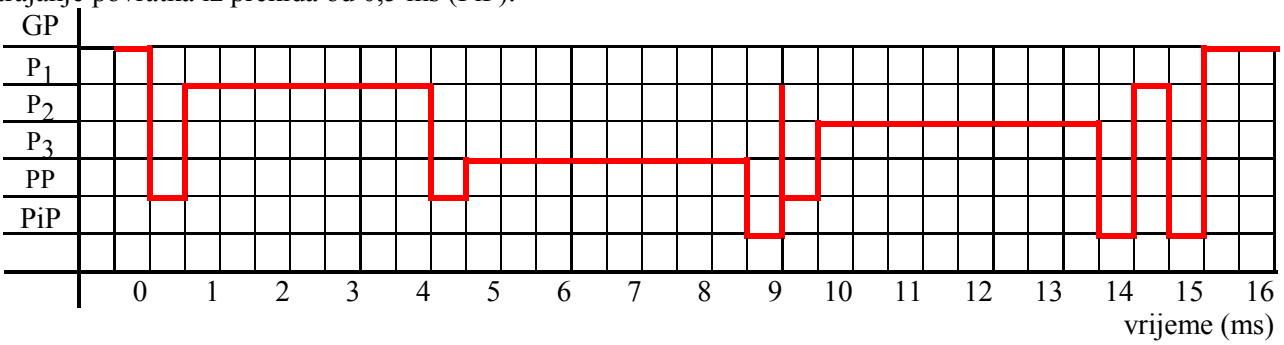
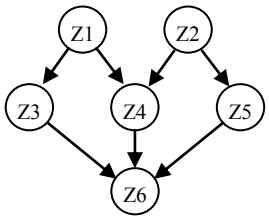
ZEMRIS, 3.2.2015.	Ime i prezime	JMBAG
Operacijski sustavi, završni ispit		

Napomena: Sve zadatke osim 4. i 11. rješavati na ovom obrascu!

1. (2) U nekom sustavu **sa sklopom za prihvat prekida** javljaju se prekidi P1 u 0. ms, P3 u 4. ms, P2 se javlja u 6. ms. Prioritet prekida određen je brojem (P3 ima najveći prioritet). Obrada svakog prekida traje po 4 ms. Grafički prikazati aktivnosti procesora u glavnom programu (GP), procedurama za obradu prekida (Pi) te procedurama za prihvat prekida (PP) i povratak iz prekida (PiP) uz trajanje prihvata prekida od 0,5 ms (PP) te trajanje povratka iz prekida od 0,5 ms (PiP).



2. (2) Sinkronizirati sustav zadataka zadan grafom korištenjem općih semafora (OSEM), tj. proširiti tekst zadatka *Ti* u *T'i* potrebnim pozivima *ČekajOSEM* i *PostaviOSEM*. Navesti početne vrijednosti semafora.



T'1: T1
PostaviOSEM(1)
PostaviOSEM(2)

T'2: T2
PostaviOSEM(2)
PostaviOSEM(3)

T'3: **ČekajOSEM(1)**
T3
PostaviOSEM(4)

T'4: **ČekajOSEM(2)**
ČekajOSEM(2)
T4
PostaviOSEM(4)

T'5: **ČekajOSEM(3)**
T5
PostaviOSEM(4)

T'6: **ČekajOSEM(4)**
ČekajOSEM(4)
ČekajOSEM(4)
T6

Početne vrijednost svih semafora su nule.

3. (3) U nekom sustavu nasumično se stvaraju dretve `A` i dretve `B`. Dretve `A` za prvi dio posla `posaoAx()` trebaju samo sredstvo `X`, a za drugi dio `posaoAxy()` i sredstvo `Y`. Dretva `A` drži sredstvo `X` dok ne obavi oba dijela posla (ne otpušta `X` nakon `posaoAx()`). Dretve `B` za svoj rad (`posaoBy()`) trebaju samo sredstvo `Y`.

   a) (2) Sinkronizirati dretve korištenjem binarnih semafora (po jedan za svako sredstvo).

```
dretva A() {
        ČekajBSEM(1)
        posaoAx()
        ČekajBSEM(2)
        posaoAxy()
        PostaviBSEM(2)
        PostaviBSEM(1)
}
```

```
dretva B() {
        ČekajBSEM(2)
        posaoBy()
        PostaviBSEM(2)
}
```

Početne vrijednosti: `BSEM[1].v = 1, BSEM[2].v = 1`

   b) (1) Može li se u takvom sustavu pojaviti potpuni zastoj? Obrazložiti.

NE. Dretva B koristi samo jedno sredstvo te potpuni zastoj ne može nastati.

4. (3) Neki poslužitelj obrađuje dvije vrste poslova: normalne i kritične. Prihvat novog posla obavlja dretva `prihvat` (po programu desno). Obradu poslova radi više dretvi `radna` – čiji tekst programa je zadatak napisati.

```
dretva prihvat {
   ponavljaj {
      posao = čekaj_novi_posao()
      Uđi_u_monitor(m)
      ako je posao.tip == KRITICAN {
         dodaj_kritičan(posao)
         br_k++   (broj krit. poslova)
      } inače {
         dodaj_normalan(posao)
         br_n++   (broj norm. poslova)
      }
      Oslobodi_iz_reda_uvjeta(red)
      Izađi_iz_monitora(m)
   } do zauvijek
}
```

Kada neka `radna` dretva odabire koji će posao uzeti uvijek će najprije uzeti kritični posao, ako takav postoji. Iznimno, ako u nekom trenutku `N` ili više dretvi obrađuje kritične poslove, radna dretva koja bi mogla uzeti novi posao (bilo koji) to ne smije napraviti, već treba čekati. Stoga, svaki puta kada dretva završi obradu kritičnog posla, ona mora osloboditi sve blokirane dretve.

Za sinkronizaciju koristiti monitore.

Uzimanje poslova obaviti funkcijama `uzmi_kritičan()` i `uzmi_normalan()`(koje postoje – nije ih potrebno ostvarivati), a obradu funkcijom `obavi_posao(posao)`. Pozivanje funkcija za preuzimanje poslova mora biti međusobno isključivo (funkcije nemaju ugrađenu sinkronizaciju). *Rješavati na košuljici!*

Rješenje na kraju.

5. (3) U nekom deterministčkom sustavu poslovi P1-P4 se javljaju periodički, svakih 20 ms. P1 se javlja prvi. Slijedi P2 nakon 5 ms, P3 nakon još 2 ms te P4 nakon još 6 ms. P1 i P2 trebaju po 6 ms, a P3 i P4 po 3 ms poslužiteljskog vremena. Izračunati: $\alpha, \beta, \rho, \bar{T}, \bar{n}$. (Konačni rezultat mora biti u [s] za $\bar{T}$ i [s^{-1}] za α i β.)

Najlakše je najprije nacrtati pa onda vidjeti koji posao kada počne i završi.

Uglavnom: P1 može doći bilo kada (nije zadan odmak od početka intervala) zato se može koristiti i 0. ms.

P1: [0, 6], P2:[0+5,6+6], P3:[0+5+2,6+6+3], P4:[0+5+2+6,6+6+3+3] => T1=6, T2=7, T3=8, T4=5

$\bar{T}$**=(6+7+6+5)/4 = 26/4 = 13/2 = 6.5 ms = 0.0065 s**

$\alpha = 4 / (20\ ms) = 200\ s^{-1}$

$\bar{n} = \alpha\bar{T} = 200 * 0.0065 = 1.3$

ρ **mogu očitati iz grafa (koliko vremena u 20 ms procesor radi / 20 ms) =>** $\rho = \frac{18}{20} = 0.9$

i od tuda $\beta = \frac{\alpha}{\rho} = \frac{200}{0.9} = 222,22\ s^{-1}$

ili $\frac{1}{\beta} = \frac{\frac{1}{Tp1}+\frac{1}{Tp2}+\frac{1}{Tp3}+\frac{1}{Tp4}}{4} = \frac{6+6+3+3}{4} = \frac{9}{2}$ **ms pa** $\beta = \frac{2}{9} * 1000\ s^{-1}$ **pa onda** $\rho = \frac{\alpha}{\beta} = 0.9$

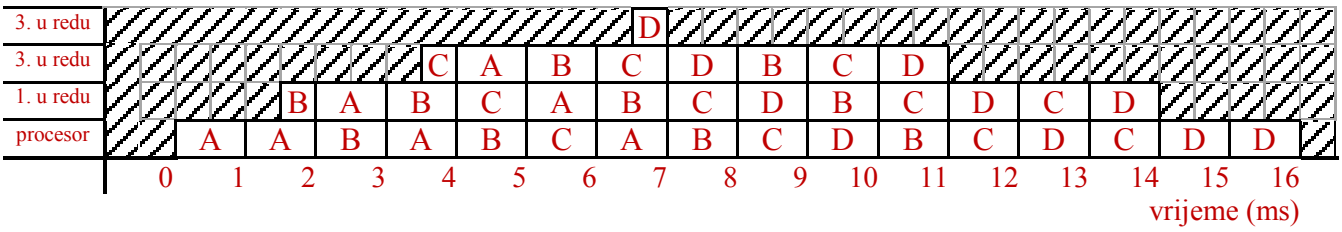
6. (2) Poslužitelj koji je radio s prosječnim opterećenjem od 0.2 (20%) zamijenjen je novim, dvostruko slabijim. Uz to dodano mu je još 50% istih poslova. Ukoliko su dolasci novih poslova modelirani Poissonovom razdiobom, a obrada eksponencijalnom, koliko će biti opterećenje novog poslužitelja?

ρ**=0.2**

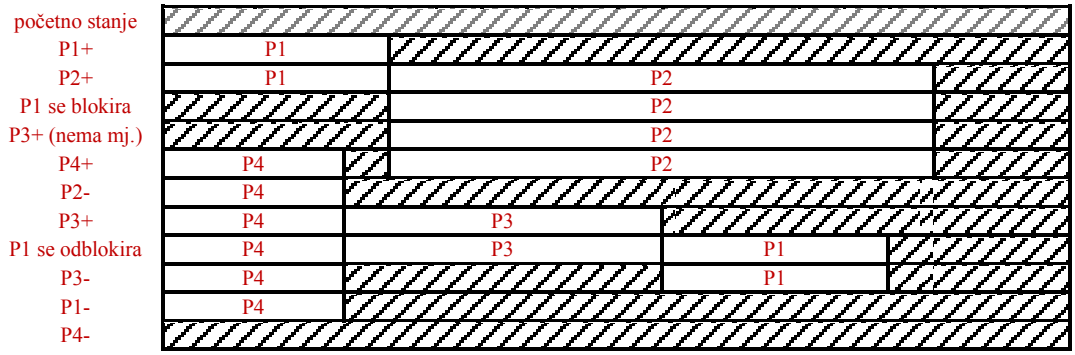
$\frac{\beta'}{\beta} = 0.5 \Rightarrow \beta' = 0.5 * \beta$ $\qquad$ $\rho' = \frac{\alpha'}{\beta'} = \frac{1.5*\alpha}{0.5*\beta} = 3\rho = 0.6$

$\frac{\alpha'}{\alpha} = 1.5 \Rightarrow \alpha' = 1.5 * \alpha$

7. (3) U nekom sustavu javljaju se dretve A, B, C, D u trenucima 0 ms, 1.5 ms, 3,5 ms i 6,5 ms, respektivno. Trajanja obrade su ista za sve dretve i iznose po 4 ms. Pokazati rad poslužitelja (grafički) ako se koristi kružno raspoređivanje sa Tq = 1 ms (kvant vremena).



8. (3) U sustavu koji koristi dinamičko upravljanje spremnikom ukupna kapaciteta 20 MB pojavljuju se sljedeći zahtjevi/događaji: pokretanje procesa P1 (koji treba 5 MB), pokretanje procesa P2 (12 MB), blokiranje procesa P1 (npr. na UI napravi), pokretanje procesa P3 (7 MB), pokretanje procesa P4 (4 MB), završetak P2, odblokiranje P1, završetak P3, završetak P1, završetak P4. Pretpostaviti da su procesi već pripremljeni na pomoćnom spremniku i da zahtjevi koji se ne mogu ostvariti u trenutku pojave čekaju i ostvaruju se kada to bude moguće (ne odbacuju se). Pokazati stanje spremnika (grafički) nakon SVAKOG događaja i SVAKE promjene u spremniku.



9. (3) Zadani algoritam (množenja) koristi kvadratne matrice dimenzija N×N i izvodi se u sustavu koji koristi straničenje s veličinom stranice od N riječi (redak matrice stane u stranicu, svaka matrica treba N stranica). Pretpostaviti da se koristi optimalna strategija zamjene stranica, te da za podatke programa (elemente svih matrica) na raspolaganju stoje tri okvira. (Dohvat instrukcija i lokalnih varijabli (i,j,k) neće izazivati promašaje.)

```
za i = 1 od N
  za k = 1 od N
    za j = 1 od N
      C[i,j] += A[i,k] * B[k,j];
```

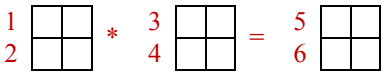
Koliko će promašaja izazvati prikazani dio programa, ako:

a) (1) N=2 (*pokazati postupak računanja*)

b) (1) N=3 (*pokazati postupak računanja*)

c) (1) općenito (za N) (*pokazati postupak ili objasniti zaključivanje o broju promašaja*)

a) N=2

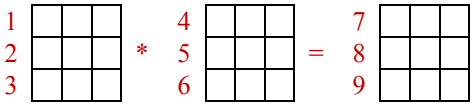


Zahtjevi: ukupno 6 stranica za sve tri matrice, zasivljeno su promašaji, tj. ima ih **8**

$\{1,3,5\}^{i=1,k=1,j=1}$, $\{1,3,5\}^{i=1,k=1,j=2}$, $\{1,4,5\}^{i=1,k=2,j=1}$, $\{1,4,5\}^{i=1,k=2,j=2}$ => 4 promašaja

$\{2,3,6\}^{i=2,k=1,j=1}$, $\{2,3,6\}^{i=2,k=1,j=2}$, $\{2,4,6\}^{i=2,k=2,j=1}$, $\{2,4,6\}^{i=2,k=2,j=2}$ => 4 promašaja

b) N=3



Zahtjevi: ukupno 9 stranica za sve tri matrice, zasivljeno su promašaji, tj. ima ih **15** = N*(N+2)

$\{1,4,7\}^{i=1,k=1,j=1..3}$, $\{1,5,7\}^{i=1,k=2,j=1..3}$, $\{1,6,7\}^{i=1,k=3,j=1..3}$ => 5 promašaja = 2+N

$\{2,4,8\}^{i=2,k=1,j=1..3}$, $\{2,5,8\}^{i=2,k=2,j=1..3}$, $\{2,6,8\}^{i=2,k=3,j=1..3}$ => 5 promašaja = 2+N

$\{3,4,9\}^{i=3,k=1,j=1..3}$, $\{3,5,9\}^{i=3,k=2,j=1..3}$, $\{3,6,9\}^{i=3,k=3,j=1..3}$ => 5 promašaja = 2+N

c) broj promašaja u općem slučaju (zaključeno na osnovu b)) = **N*(N+2)**

10. (3) Operacijski sustav treba učitati dvije datoteke velike po 4 MB u radni spremnik. Koliko će mu vremena za to trebati ako su datoteke kompaktno smještene na disku (ali svaka na svom dijelu diska, udaljene jedna od druge) te ako su svojstva diska: dvije obostrano magnetizirane ploče (4 glave), 512 staza po površini, 1024 sektora po stazi, veličina sektora je 512 B, 6000 okretaja u minuti, prijenos cijele staze u radni spremnik traje Tp=5 ms, prosječno postavljanje glave traje 10 ms, a premještanje na susjednu stazu 1 ms.

4 MB / 512 B = 4*1024*1024/512 = 8*1024 sektora treba za svaku datoteku
1 staza ima 512 sektora => 1 cilindar ima 4 staze => 4*512 sektora = 2*1024 sektora
svaka datoteka treba 8*1024/(2*1024) = 4 cilindra (puna, svaki sa po 4 staze)

operacije: postavi glavu na početak prve datoteke (Tseek)+(postavi glavu na početak staze ($\overline{T}_R$) + pročitaj cijelu stazu (T_R) + prenesi ju u radni spremnik (T_P))
prije pomicanja glave na susjedni cilindar čitaju se sve četiri staze cilindra tako da zadnja zagrada se množi sa 4
prijenosom zadnje staze (4.) u radni spremnik glava se pomiče na susjednu stazu, ali to traje kraće od prijenosa, pa nije potrebno dodavati
gornje se ponavlja za svaki cilindar tj. još jednom množi sa 4.
potom ide pomak glave na početni cilindar druge datoteke (Tseek). Ova operacija ide paralelno s prijenosom (T_P) zadnje staze (16.) prve datoteke. Ali obzirom da traje više (Tseek>T_P) razliku od Tseek-T_P treba dodati sumi. Na kraju ide isto vrijeme čitanja (4*4*zagrade)

T_R=1/(6000/60) = 10 ms => $\overline{T}_R$=5ms
T_{seek} = 10 ms
T_1= 1ms => manje od T_P pa se ne uzima u sumi
$t_Č$= Tseek + ($\overline{T}_R$ + T_R + T_P)*4*4 + Tseek-T_P + ($\overline{T}_R$ + T_R + T_P)*4*4 = 10*2+(5+10+5)*4*4*2-5 = 655 ms
$t_Č$= 655 ms = 0.655 s

11. (3) Datoteka veličine 3 MB pohranjena je na UNIX datotečnom sustavu. Veličina bloka je 2 kB, a veličina kazaljke 64 bita. Skicirati organizaciju kazaljki koje opisuju tu datoteku. *Rješavati na košuljici!*

Rješenje na kraju.

12.(10) (*teorijska pitanja – odgovarati kratko*)

(1) Što se zbiva pri izvođenju instrukcije za poziv potprograma?

programsko brojilo se pohranjuje na stog, a potom adresa potprograma se stavlja u programsko brojilo

(1) Navesti osnovne registre pristupnog sklopa za neposredni pristup spremniku (DMA).
BR-brojilo podataka, AR-adresni registar, PR-podatkovni registar, RS-registar stanja (može i bez opisa)
(1) Koji je najveći zajednički nedostatak algoritmima međusobnog isključivanja (Dekkerov, Petersonov, Lamportov te algoritmima ostvarenima uz pomoć sklopovske potpore (TAS, SWP, …))?

radno čekanje

(1) Na što se svodi izlazak iz jezgre?

obnovu konteksta aktivne dretve (+dozvola prekidanja, ali prihvatiti i samo prvi dio)

(1) Za sinkronizaciju više proizvođača s jednim potrošačem potrebno je **2** općih i **1** binarni/ih semafor/a.
(0.5) Navesti Littleovo pravilo.

$\bar{n} = \alpha \times \bar{T}$

(0.5) Što predstavlja vrijednost β u sustavu poslužitelja?

prosječan broj poslova koje poslužitelj može obraditi u jedinici vremena

(1) Čime je određena jedinstvena adresa svakog sektora na disku?

CHS: identifikacijom staze na ploči (C), identifikatorom ploče (glave/H), rednim brojem sektora na stazi (S)

(0.5) Navesti vrste fragmentacije prilikom statičkog dodjeljivanja spremnika.

unutarnja i vanjska

(0.5) Navesti Knuthovo 50% pravilo.

broj rupa jednak je 50% broja punih blokova, tj. 1/3 blokova su rupe

(1) Koju informaciju nosi bit čistoće? Gdje se on nalazi?

bit čistoće kaže je li stranica u spremniku identična njenoj kopiji na pomoćnom spremniku.
nalazi se u opisniku stranice

(1) Navesti sadržaj datotečne tablice.

Datotečna tablica sadrži opisnike datoteka.
(Sadrži i dodatne podatke o disku – opisnik particije)

Rješenje 4. zadatak

zadano:

```
dretva prihvat {
   ponavljaj {
      posao = čekaj_novi_posao()
      Uđi_u_monitor(m)
      ako je posao.tip == KRITICAN {
         dodaj_kritičan(posao)
         br_k++   (broj krit. poslova)
      } inače {
         dodaj_normalan(posao)
         br_n++   (broj norm. poslova)
      }
      Oslobodi_iz_reda_uvjeta(red)
      Izađi_iz_monitora(m)
   } do zauvijek
}
```

Početne vrijednosti varijabli su nule

Jedno od rješenja

```
dretva radna {
  ponavljaj {
    Udi_u_monitor(m) //ovaj Udi* može biti i izvan „pon.“
    dok je ( br_k+br_n == 0 || kr_dr >= N )
       Čekaj_u_redu_uvjeta ( red, m )
    ako je ( br_k > 0 ) {
       posao = uzmi_kritičan()
       br_k--
       kr_dr++
    }
    inače {
       posao = uzmi_normalan()
       br_n--
    }

    Izadi_iz_monitora(m)
    obavi_posao ( posao )
    Udi_u_monitor(m)

    ako je ( posao.tip == KRITICAN ) {
      kr_dr--
      Oslobodi_sve_iz_reda_uvjeta ( red )
    }
    Izadi_iz_monitora(m) //ovaj Izadi* može biti i izvan
  }
}
```

drugo jednako dobro rješenje

```
dretva radna {
  Udi_u_monitor(m)
  ponavljaj {
    dok je ( br_k+br_n == 0 || kr_dr >= N )
       Čekaj_u_redu_uvjeta ( red, m )
    ako je ( br_k > 0 ) {
       posao = uzmi_kritičan()
       br_k--
       kr_dr++
       Izadi_iz_monitora(m)
       obavi_posao ( posao )
       Udi_u_monitor(m)
       kr_dr--
    }
    inače {
       posao = uzmi_normalan()
       br_n--
       Izadi_iz_monitora(m)
       obavi_posao ( posao )
       Udi_u_monitor(m)
    }
  }
  Izadi_iz_monitora(m)
}
```

Rješenje 11. zadatka

blok => 2 KB

kazaljka => 64 bita => 8 B => 2 KB / 8 B = 256 kazaljki stane u jedan blok

dat. vel. 3 MB => 3*1024 KB / 2 KB = 3 * 512 blokova = 1536 blokova

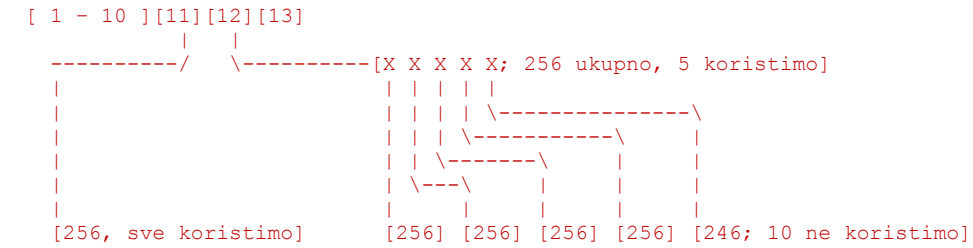
opisnik: [1 – 10][11][12][13]

1-10 => prvih 10 blokova, ostane 1526

11. pokazuje na blok s 256 kazaljki: ostane 1526 - 256 = 1270 = 256*4+246

12. kazaljka pokazuje na blok od kojeg koristimo 5 kazaljki koje pokazuju na blokove s kazaljkama

Slika kazaljki:



datoteka: [1-10] [256] [256][256][256][256][246] (opcionalno ovo može biti povezano s gornjim)

U uglatim zagradama (tj. blokovima) ne moraju biti brojke, ali mora ih biti jednako i jednako moraju biti povezane
(11->1., 12.->2., 12'->3., 4., 5., 6. i 7.)

Ispod svega može biti i logički prikaz datoteke (blokovi od 1 do 1536) i veza između blokova s kazaljkama i dijelovima datoteke.