Drugi kolokvij iz predmeta *Operacijski sustavi*, 27. 6. 2016.

1. (2) Kako se ostvaruju instrukcije "za skok" (npr. `JMP petlja`), "za poziv potprograma" (npr. `CALL funkcija`), "za povratak iz potprograma" (npr. `RET`)?

```
JMP petlja    :: PC = petlja
CALL funkcija :: PC => na stog; PC = funkcija
RET           :: sa stoga => PC
```

2. (2) Navesti nedostatke statičkog upravljanja spremnikom.

```
- fragmentacija: unutarnja i vanjska
- nema zaštite: greška jednog procesa može utjecati i na druge
- ne mogu se pokretati 'veliki' programi
```

3. (2) U sustavu koji koristi dinamičko upravljanje spremnikom, proces pozove jezgrinu funkciju i kao parametar joj pošalje adresu varijable x. Što treba napraviti u jezgrinoj funkciji da bi se moglo u tu varijablu postaviti neka vrijednost?

```
na logičku adresu treba 'ručno' zbrojiti početnu adresu procesa (fizičku
adresu početka)
jezgra ne koristi sklop - pri prelasku u jezgrin način rada sklop se
isključuje (ili se u registar početne adrese upisuje se nula, a u registar
najveće adrese najveća moguća adresa na tom sustavu i na taj način su
logičke i fizičke adrese jednake)
```

4. (2) Osim sadržaja datoteka i praznih blokova, što se sve još nalazi na disku: što je dodatno potrebno za datotečni sustav, što koristi sam disk?

```
 - datotečna tablica s opisnicima datoteka
 - opis slobodnog prostora
[- opis pokvarenih sektora]
 - na disku, uz svaki se sektor nalaze i dodatni bitovi
   (osim onih koji čine podatke sektora) koji se koriste pri detekciji i
   otklanjanju grešaka pri čitanju
```

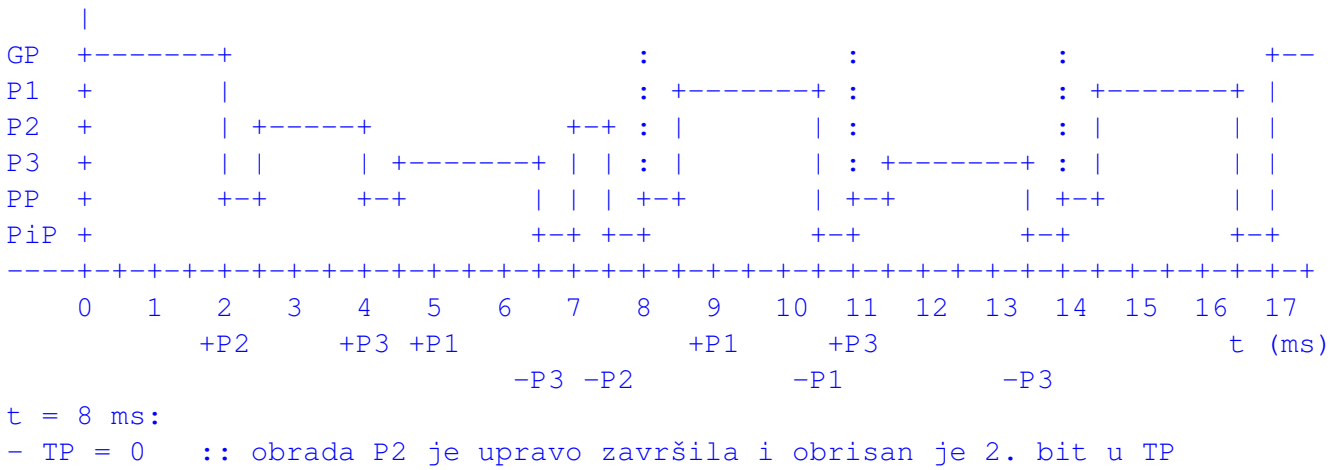
5. (2) Pri korištenju datotečna sustava, operacijski sustav mora u radnom spremniku napraviti kopiju nekih podataka. Koji su to podaci? Kada se radi njihova kopija?

```
- kopija (dijela) datotečne tablice - učitava se pri inicijalizaciji sustava
- kopija (i proširenje) opisnika datoteke - kada se datoteka otvori (od
strane programa)
- kopija blokova datoteke - kada se datoteka koristi (čita ili mijenja ili
zapisuje)
```

6. (3) Zašto je danas izuzetno bitno da se pri prisanju programa koristi uobičajeno formatiranje za dani programski jezik (uvlačenja, razmaci, zagrade, prikladno komentiranje koda)?

```
Vrlo rijetko jedan programer kreće "od nule" pri izradi nekog programa/
projekta. Uglavnom nadograđuje neki postojeći. Također, njegov kod će
koristiti drugi kada ga budu nadograđivali. Korištenjem uobičajena
formatiranja (i prikadnih komentara) će taj kod biti upotrebljiv i drugima.
```

7. U nekom sustavu javljaju se prekidi P_1 u 5. i 9. ms, P_2 u 2. ms te P_3 u 4. i 11. ms. Prioritet prekida određen je brojem (P_3 ima najveći prioritet). Sustav ima sklop za prihvat prekida. Procedura za prihvat prekida (PP) traje 0,5 ms a procedura za povratak iz prekida (PiP) 0,5 ms. Obrada svakog prekida traje po 2 ms.
   a) (3) Grafički prikazati aktivnosti procesora u glavnom programu (GP), procedurama za obradu prekida (P_i) te procedurama za prihvat prekida (PP) i povratak iz prekida (PiP).
   b) (1) Opisati stanje sustava u trenutku t=8 ms (što procesor radi, vrijednosti varijabli koji opisuju prekide, vrijednosti u registrima sklopa za prihvat prekida ...).



t = 8 ms:
- TP = 0 :: obrada P2 je upravo završila i obrisan je 2. bit u TP
- KZ = 001 :: zahtjev od P1 još nije prihvaćen (biti će u idućih 0,5 ms)

8. (2) Neki pristupni sklop koristi izravni pristup spremniku (DMA) za prijenos pristiglih podataka u radni spremnik. Trajanje jednog sabirničkog ciklusa jest 10 ns. Obrada prekida (na kraju prijenosa 10000 podataka) traži 1000 sabirničkih ciklusa (uključujući sve, od prihvata, obrade do povratka iz prekida). Uz pretpostavku da jedan podatak prosječno dolazi svakih 100 μs koji postotak procesorskog vremena se potroši na taj sklop? Pretpostaviti da će procesor izgubiti jedan takt ustupajući sabirnicu na jedan sabirnički ciklus.

- za jedan "veliki" ciklus prenosi se 10000 podataka
- jedan podatak svakih 100 us

=> 10000 * 100 us = 1 s
- u 1 sekundi ima 1 s / 10 ns = 100 000 000 sabirničkih ciklusa
- na sklop se potroši: 10000 za prijenos te 1000 za obradu prekida => 11000

=> 11000 / 100 000 000 * 100 % = 11 / 1000 = 0,011 %

9. (3) Jedan proizvođač i jedan potrošak komuniciraju preko zajedničkog međuspremnika M kapaciteta N poruka. Proizvođač stvara novu poruku sa `poruka = nova_poruka()` dok potrošač troši preuzetu poruku funkcijom `obrada(poruka)`. Napisati pseudokod proizvođača i potrošača. Za sinkronizaciju koristiti semafore. Navesti početne vrijednosti semafora i korištenih varijabli.

```
proizvođač
{
    ponavljaj {
        P = stvori poruku ();
        Čekaj_OSEM(2);
        M[ULAZ] = P;
        ULAZ = (ULAZ + 1) MOD N;
        Postavi_OSEM(1);
    }
    do zauvijek
}
```

```
potrošač
{
    ponavljaj {
        Čekaj_OSEM(1);
        R = M[IZLAZ];
        IZLAZ = (IZLAZ + 1) MOD N;
        Postavi_OSEM(2);
        obradi poruku (R);
    }
    do zauvijek
}
```

Početne vrijednosti: ULAZ = IZLAZ = 0, OSEM[2].v = N, OSEM[1].v = 0

10. (4) Svaka dretva (od njih N) "baca" kocku, pričeka da i sve ostale bace te ona s najvećim brojem se proglašava pobjednicom te runde (ispisuje se `"Dretva x je pobjednica s brojem y"`). Tek nakon toga kreće idući krug bacanja. Svaka dretva `I` ima svoju kocku pa ju može nezavisno bacati (`broj[I] = baci_kocku()`).

```
dretva (I){
  ponavljaj {
    broj[I] = baci_kocku()
    Uđi_u_monitor(m)
    ako je ( broj[I] > najveci ) tada
      najveci = broj[I]
      indeks = I
    bacilo_kocku++
    ako je ( bacilo_kocku < N ) tada
      Čekaj_u_redu_uvjeta ( red, m )
    inače //zadnja ih sve propušta dalje
      bacilo_kocku = 0
      ispisi ( "Dretva ", indeks, " je pobjednica s brojem ", najveci )
      najveci = -1
      Oslobodi_sve_iz_reda_uvjeta ( red )
    Izađi_iz_monitora(m)
  }
}
```

11. (3) U nekom sustavu postoje četiri dretve `ulazna`, `radna1`, `radna2` te `izlazna`, zadane pseudokodom:

```
ulazna() {
  ponavljaj {
    u1,u2 = dohvati()
    r1 = u1
    r2 = u2
  }
}
izlazna() {
  ponavljaj {
    z1 = i1
    z2 = i2
    spremi(z1, z2)
  }
}
```

```
radna1() {
  ponavljaj {
    x1 = r1
    y1 = obradi(x1)
    i1 = y1
  }
}
radna2() {
  ponavljaj {
    x2 = r2
    y2 = obradi(x2)
    i2 = y2
  }
}
```

Dretve razmijenjuju podatke preko zajedničkih varijabli `r1`, `r2`, `i1` i `i2`. Korištenjem binarnih semafora sinkronizirati dretve tako da se razmjena podataka zaštiti (npr. dretva `radna1` treba pričekati da dretva `ulazna` stavi podatak u `r1`; `ulazna` treba pričekati da `radna1` preuzme podatak iz `r1` prije nego li će staviti novi; ...). Sinkronizacija ne smije sprječavati paralelni rad dretvi u operacijama `dohvati()`, `obradi()` te `spremi()`. Napisati početne vrijednosti semafora.

```
ulazna() {
  ponavljaj {
    u1,u2 = dohvati()

    ČekajBSEM(1)
    r1 = u1
    PostaviBSEM(2)

    ČekajBSEM(3)
    r2 = u2
    PostaviBSEM(4)
  }
}
izlazna() {
  ponavljaj {
    ČekajBSEM(6)
    z1 = i1
    PostaviBSEM(5)

    ČekajBSEM(8)
    z2 = i2
    PostaviBSEM(7)

    spremi(z1, z2)
  }
}
```

```
radna1() {
  ponavljaj {
    ČekajBSEM(2)
    x1 = r1
    PostaviBSEM(1)

    y1 = obradi(x1)

    ČekajBSEM(5)
    i1 = y1
    PostaviBSEM(6)
  }
}
radna2() {
  ponavljaj {
    ČekajBSEM(4)
    x2 = r2
    PostaviBSEM(3)

    y2 = obradi(x2)

    ČekajBSEM(7)
    i2 = y2
    PostaviBSEM(8)
  }
}
```

```
Početne vrijednosti semafora:
- semafori 1, 3, 5, 7 => .v = 1
- semafori 2, 4, 6, 8 => .v = 0
```

12. (3) U nekom sustavu javljaju se dretve A, B, C i D. Dretva A ima najveći prioritet, slijede dretve B i C koje imaju jednaki prioritet, te dretva D koja ima najmanji prioritet. Dretva A javlja se u t=4 s, B u t=2 s, C u t=5 s te D u t=1 s. Svaka dretva treba 5 s procesorskog vremena. Sustav koristi raspoređivanje `SCHED_RR` (prioritet pa podjela vremena). Prikazati rad sustava.

```
3.pr.+          |D|D|D|D|
2.pr.+        |D|C|C|C|C|D|D|D|D|D|
1.pr.+    |D|D|B|B|B|B|B|C|B|C|B|C|D|D|D|
proc + |D|B|B|A|A|A|A|A|B|C|B|C|B|C|C|C|D|D|D|D|
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
        +D+B   +A+C       -A          -B     -C       -D
```

13. (5) U nekom sustavu koji koristi dinamičko upravljanje spremnikom u nekom periodu javlja se pet zahtjeva za pokretanjem programa, svakih 100 ms jedan. Neka su zahtjevi za spremnikom tih programa (P1-P5) redom: 10 MB, 20 MB, 10 MB, 30 MB i 5 MB (zadnji zahtjev). Svaki program treba 200 [ms] procesorskog vremena. Neka se trajanje učitavanja s pomoćnog spremnika može izračunti prema veličini programa: 2*veličina [ms] (kada je veličina u MB). Sustav ima za programe na raspolaganju 35 MB spremničkog prostora. Procesor ima 4 jezgre (4 programa bi mogao izvoditi paralelno). Prikazati stanje sustava (što je u radnom spremniku, što na kojoj jezgri procesora) dok se svih pet poslova ne završi. Svi procesi su jednako bitni. Jednom započeti proces neće se prekidati (neće se izbaciti iz glavnog u pomoćni spremnik).

```
    |          spremnik          |
t   |   |   |   |   |   |   |   | komentar
----+----------------------------+-----------------------------------------
0   |                            | početno stanje (može i ovdje sve početi)
----+----------------------------+-----------------------------------------
100 |xxxxxxx|                    | učitavanje P1
----+----------------------------+-----------------------------------------
120 |   P1  |                    | P1 radi
----+----------------------------+-----------------------------------------
200 |   P1  |xxxxxxxxxxxxxxx|    | P1 radi, P2 se učitava
----+----------------------------+-----------------------------------------
240 |   P1  |       P2      |    | P1 radi, P2 radi
----+----------------------------+-----------------------------------------
300 |   P1  |       P2      |    | P1 radi, P2 radi, zahtjev za P3 čeka
----+----------------------------+-----------------------------------------
320 |xxxxxxx|       P2      |    | P1 završio, P2 radi, P3 se učitava
----+----------------------------+-----------------------------------------
360 |   P3  |       P2      |    | P2 radi, P3 radi
----+----------------------------+-----------------------------------------
400 |   P3  |       P2      |    | P2 radi, P3 radi, zahtjev za P4 čeka
----+----------------------------+-----------------------------------------
440 |   P3  |                    | P2 završio, P3 radi, zahtjev za P4 čeka
----+----------------------------+-----------------------------------------
500 |   P3  |xxx|                | P3 radi, zahtjev za P4 čeka, P5 se učitava
----+----------------------------+-----------------------------------------
510 |   P3  | P5|                | P3 radi, P5 radi, zahtjev za P4 čeka
----+----------------------------+-----------------------------------------
560 |       | P5|                | P3 završio, P5 radi, zahtjev za P4 čeka
----+----------------------------+-----------------------------------------
710 |xxxxxxxxxxxxxxxxxxxxxxx|    | P5 završio, P4 se učitava
----+----------------------------+-----------------------------------------
770 |           P4          |    | P4 radi
----+----------------------------+-----------------------------------------
970 |                            | P4 završio
----+----------------------------+-----------------------------------------
```

14. (3) Zadatak je nekog program zbrojiti dvije ogromne matrice A i B, uz spremanje rezultata u matricu B (B = A + B). Matrice su kvadratne, dimenzija $N \times N$. Sustav koristi straničenje s veličinom stranice V za koju vrijedi $N = 4 \times V$. Ukoliko za program stoje na raspolaganju 2 okvira (za matrice, zanemariti ostale potrebe programa), koliko će promašaja izazvati program pri zbrajanju tih matrica? Korišteni algoritam zamjena stranica je LRU (engl. *least recently used*).

```
- pretpostavljeni algoritam:

   za i = 1 do N
      za j = 1 do N
         B[i,j] = A[i,j] + B[i,j]

- svaki redak u 4 stranice => svaka matrica ima 4N stranica => ukupno 8N
- zbrajanje je "savršeno" sljedno:
  - paralelno se koristi jedna stranica od A i jedna stranica od B u
    potpunosti prije nego li se počnu koristiti iduće
  - nema suvišnih promašaja (svaka se stranica učita samo jednom)
- ukupan broj promašaja = 8N
```

15. (2) Neka datoteka zauzima 100 blokova i kompaktno je smještena na disku, počevši od bloka diska 123457. Opisati smještaj te datoteke ako se koristi NTFS sustav.

```
VCN | LCN    | #
----+--------+-----
 1  | 123457 | 100
```