Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

*Predmet:* Operativni sistemi 1 (SI2OS1, IR2OS1)
*Nastavnik:* prof. dr Dragan Milićev
*Odsek:* Softversko inženjerstvo, Računarska tehnika i informatika
*Kolokvijum:* Prvi, septembar 2016.
*Datum:* 4.9.2016.

# *Prvi kolokvijum iz Operativnih sistema 1*

*Kandidat:* _____

*Broj indeksa:* _____ *E-mail:*_____

*Kolokvijum traje 1,5 sat. Dozvoljeno je korišćenje literature.*

*Zadatak 1* _____/10 *Zadatak 3* _____/10
*Zadatak 2* _____/10

**Ukupno:** _____/30 = _____% = _____/15

**Napomena:** Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

## 1. (10 poena)

Date su deklaracije pokazivača preko kojih se može pristupiti registrima jednog kontrolera izlaznog uređaja:

```
typedef unsigned int REG;
REG* ioCtrl =...;     // Output device control register
REG* ioData =...;     // Output device data register
const int BUFSIZE = ..., DELAY = ...;
REG buffer[BUFSIZE];
```

U upravljačkom registru najniži bit je bit *Start* kojim se pokreće uređaj. Statusni registar nije programski dostupan, niti kontroler signalizira spremnost na bilo koji način (npr. prekidom), ali je poznato da će, nakon zadavanja prenosa jednog podatka, kontroler sigurno završiti operaciju i biti spreman za novu nakon `DELAY` vremena.

Na jeziku C napisati kod operacije `transfer()` koja vrši prenos bloka podataka iz bafera `buffer` na dati izlazni uređaj. Ova operacija poziva se iz konteksta kernel niti. U kernelu je dostupna operacija `sleep(int)` koja uspavljuje (suspenduje) pozivajuću nit na zadato vreme. Sve eventualne greške ignorisati.

Rešenje:

## 2. (10 poena)

Školsko jezgro proširuje se konceptom tzv. *tačke spajanja* ili *susreta* (engl. *join, rendez-vous*), kao jednostavne sinhronizacione primitive sa sledećim značenjem.

Program može kreirati objekat klase `Join`, čiji je interfejs dat dole, zadajući mu kao parametre konstruktora (pokazivače na) dve niti koje će se sinhronizovati (susretati, „spajati“) na ovom objektu. Iz konteksta ove dve niti (i samo njih) može se pozvati operacija `wait` ovog objekta. Tom operacijom se ove dve niti „spajaju“ (sinhronizuju, susreću), odnosno međusobno sačekuju, tako da ni jedna od njih neće nastaviti svoje izvšavanje dalje ako ona druga nije stigla do iste te tačke spajanja (tj. pozvala ovu operaciju `wait` na istom ovom objektu klase `Join`). Drugim rečima, niti nastavljaju izvršavanje iza poziva `wait` samo kada su obe stigle do te tačke.

Dole je data implementacija sistemskog poziva `dispatch()` u školskom jezgru, kao primer kako su implementirana sva mesta promene konteksta. Implementirati u celini klasu `Join`, bez ikakve izmene ili dopune klase `Thread`. Operacija `wait` treba da vrati sledeće:

- -1 u slučaju da je ovu operaciju pozvala neka druga nit, osim one dve koje su definisane konstruktorom (ili bilo kakve druge greške);
- 0 u slučaju da je pozivajuća nit prva stigla do tačke spajanja, a 1 u suprotnom.

```
void dispatch () {
  lock();
  jmp_buf old = Thread::running->context;
  Scheduler::put(Thread::running);
  Thread::running = Scheduler::get();
  jmp_buf new = Thread::running->context;
  if (setjmp(old)==0) longjmp(new,1);
  unlock();
}

class Join {
public:
  Join (Thread* thread1, Thread* thread2);
  int wait ();
};
```

Rešenje:

**3. (10 poena)**

Dat je neki veliki jednodimenzioni niz `data` veličine `N`, čiji su elementi tipa `Data` (deklaracije dole). Jedan element niza obrađuje procedura `processData`. Ovaj niz potrebno je obraditi pomoću `n` uporednih niti (procedura `parallelProcessing`), gde je `n` zadati parametar, tako što svaka od tih uporednih niti iterativno obrađuje približno isti broj elemenata ovog velikog niza. Drugim rečima, niz treba particionisati na `n` disjunktnih podnizova, što približnije jednakih, i te particije obrađivati uporedo. Implementaciju dati u školskom jezgru.

```
const int N = ...;
class Data;
Data data[N];
void processData(Data*);
void parallelProcessing (int n);
```

Rešenje: