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 2012.
Datum: 31.8.2012.

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	*Zadatak 4*	_____/10

Ukupno: _____/40 = _____% = _____/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 dva ulazna uređaja:

```
typedef unsigned int REG;
REG* io1Ctrl =...;   // Device 1 control register
REG* io1Status =...; // Device 1 status register
REG* io1Data =...;   // Device 1 data register
REG* io2Ctrl =...;   // Device 2 control register
REG* io2Status =...; // Device 2 status register
REG* io2Data =...;   // Device 2 data register
```

U upravljačkim registrima najniži bit je bit *Start* kojim se pokreće uređaj, a u statusnim registrima najniži bit je bit spremnosti (*Ready*), a bit do njega bit greške (*Error*). Svi registri su veličine jedne mašinske reči (tip `unsigned int`).

Potrebno je napisati kod koji vrši prenos po jednog bloka podataka sa svakog od dva uređaja uporedo, sa prvog uređaja tehnikom prozivanja (*polling*), a sa drugog korišćenjem prekida. U slučaju greške na bilo kom uređaju, prenos sa tog uređaja treba prekinuti, a sa drugog uređaja treba nastaviti (osim ukoliko je i kod njega došlo do greške). Transfer se obavlja pozivom sledeće funkcije iz koje se vraća kada su oba prenosa završena:

```
int transfer (unsigned* blk1, int count1, unsigned* blk2, int count2);
```

Povratna vrednost ove funkcije treba da bude ceo broj čija binarna predstava u najniža dva bita ima sledeće značenje: bit 0 ukazuje na grešku u prenosu sa prvog uređaja (0-ispravno, 1-greška), a bit 1 na grešku u prenosu sa drugog uređaja.

Rešenje:

2. (10 poena)

Virtuelna memorija nekog računara organizovana je stranično. Veličina virtuelnog adresnog prostora je 16 MB, adresibilna jedinica je bajt, a veličina stranice je 64 KB. Veličina fizičkog adresnog prostora je 16 MB. Operativni sistem učitava stranice na zahtev, tako što se stranica učitava u prvi slobodni okvir fizičke memorije kada joj se pristupi. Kada se kreira proces, ni jedna njegova stranica se ne učitava odmah, već tek kad joj se prvi put pristupi. U početnom trenutku, slobodni okviri fizičke memorije su okviri počev od 20h zaključno sa 2Fh. Neki proces generiše sledeću sekvencu virtuelnih adresa tokom svog izvršavanja (sve vrednosti su heksadecimalne):

30F00, 30F02, 30F04, 822F0, 822F2, 322F0, 322F2, 322F4, 522F0, 522F2, 602F0, 602F2

Prikazati izgled prvih 16 ulaza tabele preslikavanja stranica (PMT) za ovaj proces posle izvršavanja date sekvence. Za svaki ulaz u PMT prikazati indikator prisutnosti stranice u fizičkoj memoriji (0 ili 1) i broj okvira u fizičkoj memoriji u koji se stranica preslikava, ukoliko je stranica učitana; ukoliko nije, prikazati samo ovaj indikator.

Rešenje:

3. (10 poena)

U nekom sistemu realizovana je operacija

```
void yield (jmp_buf old, jmp_buf new);
```

koja čuva kontekst niti čiji je `jmp_buf` dat kao prvi argument, oduzima joj procesor i restaurira kontekst niti čiji je `jmp_buf` dat kao drugi argument, kojoj predaje procesor.

Koristeći ovu operaciju `yield()`, realizovati operacije:

- `Thread::suspend()`: (statička) suspenduje (blokira) izvršavanje pozivajuće niti sve dok je neka druga nit ne „probudi“ pomoću `resume()`;
- `Thread::resume()`: (nestatička) „budi“ (deblokira) nit za koju je pozvana; vrši i promenu konteksta predajući procesor niti koja je na redu za izvršavanje.

Pretpostaviti da je lista spremnih procesa uvek neprazna prilikom suspenzije niti i da je nit za koju se poziva `resume` sigurno suspendovana (ignorisati mogućnost greške). Klase `Thread` i `Scheduler` su u preostalim delovima implementirane kao u školskom jezgru.

Rešenje:

4. (10 poena)

U nekom sistemu niti se kreiraju sistemskim pozivom `fork()` koji ima istu sintaksu i značenje povratne vrednosti kao i istoimeni Unix sistemski poziv, samo što umesto procesa kreira nit u istom adresnom prostoru roditelja: novokreirana nit ima istu poziciju u izvršavanju kao i roditeljska nit, deli isti adresni prostor, samo što poseduje sopstveni kontrolni stek koji inicijalno predstavlja identičnu kopiju roditeljskog steka u trenutku poziva `fork()`. Sistemski poziv `exit()` gasi pozivajuću nit.

Korišćenjem ovih sistemskih poziva, realizovati funkciju

```
int create_thread (void (*fun)(void*), void* p);
```

koja kreira nit nad potprogramom na koji ukazuje prvi argument, tako što taj potprogram poziva u kontekstu novokreirane niti sa datim argumentom `p`, i završava kreiranu nit nakon završetka tog potprograma. Ukoliko sistemski poziv `fork()` vrati grešku, isti kod greške treba vratiti i pozivaocu funkcije `create_thread()`. Ukoliko je kreiranje niti uspešno, pozivaocu funkcije `create_thread()` treba vratiti rezultat poziva `fork()` (ID kreirane niti).

Rešenje: