Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Prvi, jun 2018.
Datum: 20. 6. 2018.

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 tri uređaja; prvi uređaja je ulazni, a druga dva su izlazni:

```
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
REG* io3Ctrl =...;   // Device 3 control register
REG* io3Status =...; // Device 3 status register
REG* io3Data =...;   // Device 3 data register
```

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

Potrebno je napisati potprogram `transfer` koji učitava po jednu reč sa ulaznog uređaja i odmah tu učitanu reč šalje na bilo koji od ova dva izlazna uređaja, koji god od njih je pre spreman. Ovaj prenos se obavlja sve dok se sa ulaznog uređaja ne učita vrednost 0. Sve prenose vršiti programiranim ulazom/izlazom sa prozivanjem (*polling*).

Rešenje:

2. (10 poena)

Neki procesor obrađuje prekide (hardverske i softverske) tako što tokom izvršavanja prekidne rutine koristi poseban stek koji se koristi u sistemskom, privilegovanom režimu rada procesora, u kome se izvršava kod kernela (čiji je deo i prekidna rutina).[1] Taj stek alociran je u delu memorije koju koristi kernel, a na vrh tog steka ukazuje poseban registar SSP procesora koji je dostupan samo u privilegovanom režimu.

Prilikom obrade prekida, procesor na ovom steku čuva: pokazivač vrha korisničkog steka (SP) koji je koristio prekinuti proces, programsku statusnu reč (PSW) i programski brojač (PC), tim redom, ali *ne* i ostale programski dostupne registre. Prilikom povratka iz prekidne rutine instrukcijom `iret`, procesor restaurira ove registre sa sistemskog steka i vraća se u korisnički režim, a time i na korisnički stek.

Svaki proces ima sopstveni takav sistemski stek. Prilikom promene konteksta, ostale programski dostupne registre treba sačuvati na ovom sistemskom steku prekinutog procesa. U strukturi PCB postoji polje za čuvanje vrednosti SSP steka procesa; pomeraj ovog polja u odnosu na početak strukture PCB označen je simboličkom konstantom `offsSSP`.

Procesor je RISC, sa *load/store* arhitekturom i ima sledeće programski dostupne registre: 32 registra opšte namene (R0..R31), SP, PSW i PC.

Sistem je multiprocesorski. Poseban registar procesora RPID služi za identifikaciju tekućeg procesa koji taj procesor trenutno izvršava i dostupan je samo u privilegovanom režimu. Potprogram `scheduler`, koji nema argumente, realizuje raspoređivanje, tako što smešta adresu PCB onog procesa koji se raspoređuje na datom procesoru (na kome se izvršava kod) u registar RPID.

a)(7) Na asembleru datog procesora napisati kod prekidne rutine `dispatch` koja vrši promenu konteksta korišćenjem potprograma `scheduler`. Ova prekidna rutina poziva se sistemskim pozivom iz korisničkog procesa, pri čemu se sistemski poziv realizuje softverskim prekidom.

b)(3) Pod pretpostavkom da procedura `scheduler` obezbeđuje međusobno isključenje pristupa deljenim strukturama podataka (npr. redu spremnih procesa) od strane različitih procesora, da li i ostatak prekidne rutine `dispatch` treba da obezbedi isto, tj. da li je u nju potrebno ugraditi *spin lock*? Precizno obrazložiti odgovor.

Rešenje:

[1] Na ovaj način rade mnogi procesori. Na primer, opis koji sledi je donekle izmenjen i pojednostavljen opis načina funkcionisanja savremenih Intel procesora sa arhitekturom IA-32 i IA-64.

3. (10 poena)

U nastavku je data donekle izmenjena i pojednostavljena specifikacija dva sistemska poziva iz standardnog *POSIX thread* API (*Pthreads*). Obe ove funkcije vraćaju negativnu vrednost u slučaju greške, a 0 u slučaju uspeha.

- `int pthread_create(pthread_t *thread, void *(*start_routine)(void *), void *arg)`: kreira novu nit nad funkcijom na koju ukazuje `start_routine`, dostavljajući joj argument `arg`; identifikator novokreirane niti smešta u loakciju na koju ukazuje `thread`;
- `int pthread_join(pthread_t thread, void **retval)`: čeka da se nit identifikovana sa `thread` završi (ukoliko se već završila, odmah vraća kontrolu); ako `retval` nije `NULL`, kopira povratnu vrednost funkcije `start_routine` koju je ta nit izvršavala u lokaciju na koju ukazuje `retval`.

Data je globalna kvadratna matrica `mat` celih brojeva (`int`), dimenzija $N \times N$. Potrebno je napisati potprogram `par_sum` koji treba da izračuna ukupan zbir svih elemenata matrice `mat`, ali na sledeći način: najpre treba da izračuna zbir svake, i-te vrste ove matrice, uporedo, u N uporednih niti (zbir jedne vrste izračunava jedna nit), i da taj zbir smesti u i-ti element jednog pomoćnog niza; potom treba da sabere ovako izračunate zbirove vrsta i vrati rezultat. Ignorisati eventualne greške i prekoračenje.

```
const int N = ...;
int mat[N][N];
int par_sum ();
```