Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

*Predmet:* Operativni sistemi 1 (SI2OS1)
*Nastavnik:* prof. dr Dragan Milićev
*Odsek:* Softversko inženjerstvo
*Kolokvijum:* Prvi, Mart 2013.
*Datum:* 24.3.2013.

# *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 uazno/izlazna 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*). Svi registri su veličine jedne mašinske reči (tip `unsigned int`).

Potrebno je napisati kod koji najpre vrši ulaz bloka podataka zadate veličine sa prvog uređaja korišćenjem prekida, a potom izlaz tog istog učitanog bloka podataka na drugi uređaj tehnikom prozivanja (*polling*).

Rešenje:

**2. (10 poena)**

Neki računar podržava segmentno-straničnu organizaciju virtuelne memorije, pri čemu je virtuelna adresa 16-bitna, fizički adresni prostor je veličine 8GB, a adresibilna jedinica je 16-bitna reč. Stranica je veličine 512B. Maksimalan broj segmenata u virtuelnom adresnom prostoru je 4. Prikazati logičku strukturu virtuelne i fizičke adrese i navesti širinu svakog polja.

Rešenje:

**3. (10 poena)**

U nekom operativnom sistemu svi sistemski pozivi izvršavaju se kao softverski prekid koji skače na prekidnu rutinu označenu kao `sys_call`, dok se sama identifikacija sistemskog poziva i njegovi parametri prenose kroz registre procesora. Jezgro tog operativnog sistema je višenitno – poseduje više internih kernel niti koje obavljaju različite poslove: izvršavaju uporedne I/O operacije, vrše druge interne poslove jezgra, pa čak postoje i niti koje obavljaju sve potrebne radnje prilikom promene konteksta korisničkih procesa (osim samog čuvanja i restauracije konteksta procesora), kao što su smeštanje PCB korisničkog procesa koji je do tada bio tekući u odgovarajući red (spremnih ili blokiranih, u zavisnosti od situacije), izbor novog tekućeg procesa iz skupa spremnih, promenu memorijskog konteksta, obradu samog konkretnog sistemskog poziva, itd. Prilikom obrade sistemskog poziva `sys_call`, prema tome, treba samo oduzeti procesor tekućem korisničkom procesu i dodeliti ga tekućoj kernel niti.

Na PCB tekućeg korisničkog procesa ukazuje globalni pokazivač `runningUserProces`, a na PCB tekuće interne niti jezgra ukazuje globalni pokazivač `runningKernelThread`. I interne niti jezgra vrše promenu konteksta između sebe na isti način, pozivom softverskog prekida koji ima istu internu strukturu kao i rutina `sys_call`.

Prilikom obrade prekida, procesor čuva ceo kontekst svog izvršavanja, odnosno sve programski dostupne registre (uključujući i PC, SP i PSW) u memoriji. Međutim, kako se ne bi dogodilo prekoračenje steka korisničkog procesa ili stranična greška prilikom tog postupka, procesor ove registre čuva u strukuturu u memoriji čija se adresa nalazi u posebnom namenskom registru procesora SPX. Prilikom povratka iz prekidne rutine, procesor sam restaurira kontekst izvršavanja iz iste ove strukture na koju tada ukazuje SPX. Ovaj registar dostupan je samo iz privilegovanog sistemskog režima rada procesora u koji se prelazi softverskim prekidom.

I korisnički procesi i interne niti jezgra u svojim PCB strukturama imaju polje pod nazivom *context* u kojima se čuva kontekst procesora. Pomeraj ove strukture u odnosu na početak PCB dat je simboličkom vrednošću `offsContext`. Procesor je RISC, sa *load/store* arhitekturom.

Upotreba imena globalne promenljive iz C koda u asembleru datog procesora predstavlja memorijsku adresu te globalne statičke promenljive. Deo memorije koju koristi jezgro (kod i podaci) preslikava se u virtuelni adresni prostor svih korisničkih procesa, tako da prilikom obrade sistemskog poziva i prelaska u kod kernela nema promene memorijskog konteksta.

Na asembleru datog procesora napisati prekidnu rutinu `sys_call`. Obrazložiti rešenje!

Rešenje:

## 4. (10 poena)

U nekom operativnom sistemu postoje sledeći sistemski pozivi:

- `int thread_create(void (*)(int), int)`: kreira nit nad funkcijom na koju ukazuje prvi argument; ta funkcija prima jedan celobrojni argument i ne vraća rezultat; novokreirana nit poziva tu funkciju sa stvarnim argumentom jednakim drugom argumentu ovog sistemskog poziva; sistemski poziv vraća PID kreirane niti;
- `void wait(int pid)`: suspenduje pozivajuću roditeljsku nit dok se ne završi nit-dete sa zadatim PID.

Data je celobrojna konstanta `N` i funkcija `f` koja prima dva celobrojna argumenta:

```
int f(int,int);
```

Na jeziku C napisati program koji u *N***N* uporednih niti-dece izračunava vrednost funkcije `f(i,j)`, `i=0,...,N-1`, `j=0,...,N-1`, svaku vrednost u po jednoj niti. Kada sve niti-deca završe, program treba da ispiše sve dobijene vrednosti redom na standardni izlaz.

Rešenje: