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 2014.
*Datum:* 23.3.2014.

# *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 jednog DMA kontrolera:

```
typedef unsigned int REG;
REG* dmaCtrl =...;    // DMA control register
REG* dmaStatus =...; // DMA status register
REG* dmaAddress =...;    // DMA block address register
REG* dmaCount =...;    // DMA block size register
```

U upravljačkom registru najniži bit je bit *Start* kojim se pokreće prenos jednog bloka preko DMA, a u statusnom registru najniži bit je bit završetka prenosa (*TransferComplete*), a bit do njega bit greške (*Error*). Svi registri su veličine jedne mašinske reči (tip `unsigned int`).

Zahtevi za ulaznim operacijama na nekom uređaju sa kog se prenos blokova vrši preko ovog DMA kontrolera vezani su u jednostruko ulančanu listu. Zahtev ima sledeću strukturu:

```
struct IORequest {
  REG* buffer; // Data buffer (block)
  unsigned int size; // Buffer (blok) size
  int status; // Status of operation
  IORequest* next; // Next in the list
};
```

Na prvi zahtev u listi pokazuje globalni pokazivač `ioHead`. Kada u praznu listu kernel stavi prvi zahtev, pozvaće operaciju `transfer()` koja treba da pokrene prenos za taj prvi zahtev. Kada se završi prenos zadat jednim zahtevom, potrebno je u polje `status` date strukture preneti status završene operacije (0 – ispravno završeno do kraja, -1 – greška), izbaciti obrađeni zahtev iz liste i pokrenuti prenos za sledeći zapis u listi. Ako zahteva u listi više nema, ne treba uraditi više ništa (kada bude stavljao novi zahtev u listu, kernel će proveriti i videti da je ona bila prazna, pa pozvati ponovo operaciju `transfer()` itd.)

Potrebno je napisati kod operacije `transfer()`, zajedno sa odgovarajućom prekidnom rutinom `dmaInterrupt()` za prekid od DMA kontrolera.

```
void transfer ();
interrupt void dmaInterrupt ();
```

Rešenje:

**2. (10 poena)**

Neki računar podržava segmentnu organizaciju virtuelne memorije, pri čemu je virtuelna adresa 16-bitna, fizički adresni prostor je veličine 4GB, a adresibilna jedinica je bajt. Maksimalna veličina segmenta je 4KB. U sledećoj tabeli dat je sadržaj nekoliko ulaza u tabeli preslikavanja nekog procesa (sve vrednosti su heksadecimnalne):

| Segment # | Size | In memory? | Starting physical address |
|---|---|---|---|
| 1 | 500 | Yes | 250000 |
| 2 | 600 | Yes | AFF00 |
| A | 700 | No | - |
| B | 900 | No | - |
| C | 800 | Yes | D6700 |

Popuniti sledeću tabelu na sledeći način: ako data virtuelna adresa uzrokuje grešku prekoračenja - napisati X, ukoliko izaziva straničnu grešku (*page fault*) - napisati P, a ukoliko je preslikavanje uspešno, upisati fizičku adresu u koju se preslikava (heksadecimalno).

| Virtual address (hex) | Mapping result (hex) |
|---|---|
| 12FA | |
| C0FF | |
| 1675 | |
| B014 | |
| CDAB | |

Rešenje:

3. **(10 poena)**

Na asembleru datog procesora napisati kod operacije

```
void yield (PCB* cur, PCB* nxt);
```

poput one date na predavanjima, a koja u PCB na koji ukazuje prvi argument čuva kontekst izvršavanja koje se napušta i restaurira kontekst koji je sačuvan u PCB na koga ukazuje drugi argument.

Procesor je RISC, sa *load/store* arhitekturom i ima sledeće programski dostupne registre: 32 registra opšte namene (R0..R31), SP, PSW i PC. Registre (PSW, R0..R31 i SP) treba sačuvati u odgovarajuća polja strukture PCB. U strukturi PCB postoje polja za čuvanje svih tih registara; pomeraji ovih polja u odnosu na početak strukture PCB označeni su simboličkim konstantama `offsPSW`, `offsSP`, `offsR0`, ..., `offsR31`.

Rešenje:

## 4. (10 poena)

U sistemu UNIX i sličnim sistemima sistemski poziv

```
int execvp(const char * filename, char * const args[]);
```

radi isto što i poziv execlp, s tim što drugi argument predstavlja niz pokazivača na nizove znakova (*null-terminated strings*) koji predstavljaju listu argumenata poziva programa koji treba izvršiti. Po konvenciji, prvi argument pokazuje na naziv fajla samog programa koji se izvršava. Ovaj niz pokazivača mora da se završi elementom sa vrednošću NULL.

U nastavku je dat jedan neispravan program. Namera programera je bila da sastavi program koji prima jedan celobrojni argument (po konvenciji, u programu se on vidi kao drugi argument, pored naziva programa), i da, samo ukoliko je vrednost celobrojnog argumenta veća od 0, kreira proces-dete nad istim programom i sa za jedan manjom vrednošću svog celobrojnog argumenta, potom ispiše svoju vrednost argumenta i sačeka da se proces-dete završi, a onda se i sam završi. Proces-dete radi to isto (i tako rekurzivno).

Napisati ispravnu implementaciju ovog programa.

```
void main (int argc, char* const argv[]){
  if (argc<2) return; // Exception!

  // Get the argument value:
  int myArg = 0;
  sscanf(argv[1],"%d",&myArg);
  if (myArg<=0) return;

  // Prepare the arguments for the child:
  char childArg[10];  // The value of the second argument
  sprintf(childArg,"%d",myArg-1);
  char* childArgs[3];
  childArgs[0] = argv[0];
  childArgs[1] = childArg;
  childArgs[2] = NULL;

  // Create a child:
  execvp(argv[0],childArgs);
  printf("%s\n",myArg);
  wait(NULL);
}
```

Rešenje: