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 2014.
Datum: 29.8.2014.

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 ulaznog uređaja i registru posebnog uređaja – vremenskog brojača:

```
typedef unsigned int REG;
REG* ioCtrl =...;   // Device control register
REG* ioData =...;   // Device data register
REG* timer = ...;   // Timer
```

Učitavanje svakog pojedinačnog podatka sa ovog ulaznog uređaja zahteva se posebnim upisom vredosti 1 u najniži bit upravljačkog registra ovog uređaja. Spremnost ulaznog podatka u registru za podatke uređaj ne signalizira nikakvim signalom, ali je sigurno da je ulazni podatak spreman u registru podataka najkasnije 50 ms nakon zadatog zahteva (upisa u kontrolni registar).

Na magistralu računara vezan je i registar posebnog uređaja, vremenskog brojača. Upisom celobrojne vrednosti *n* u ovaj registar, vremenski brojač počinje merenje vremena od *n* ms i, nakon isteka tog vremena, generiše prekid procesoru.

Na jeziku C napisati kod operacije `transfer()` zajedno sa odgovarajućom prekidnom rutinom za prekid od vremenskog brojača `timerInterrupt()`, koja obavlja učitavanje bloka podataka zadate dužine na zadatu adresu u memoriji sa datog ulaznog uređaja.

```
void transfer (REG* buffer, unsigned int count);
interrupt void timerInterrupt ();
```

Rešenje:

2. (10 poena)

Neki računar podržava segmentno-straničnu organizaciju virtuelne memorije, pri čemu je virtuelni adresni prostor veličine 16GB, adresibilna jedinica je 32-bitna reč, a fizički adresni prostor je veličine 1GB. Maksimalna veličina segmenta je 64MB, a stranica je veličine 1KB.

a)(5) Prikazati logičku strukturu virtuelne i fizičke adrese i označiti širinu svakog polja.

b)(5) Napisati heksadecimalni kod fizičke adrese u koju se preslikava virtuelna adresa u segmentu 56h, stranici broj 34h u tom segementu, reč broj DDh u toj stranici, ako se ta stranica preslikava u okvir broj FF00h.

3. (10 poena)

Neki troadresni RISC procesor sa *load/store* arhitekturom, poput onog opisanog na predavanjima, poseduje 32 registra opšte namene, označenih sa *R*0..*R*31, statusnu reč PSW i pokazivač steka SP, koji su dostupni instrukcijama koje se izvršavaju u korisničkom režimu rada procesora, kao i dva posebna registra *Rx* i *Rp* koji su dostupni samo u privilegovanom (sistemskom) režimu rada procesora. Registar *Rx* se može koristiti kao i bilo koji registar *R*0..*R*31, i operativni sistem ga može koristiti proizvoljno za sopstvene potrebe (npr. prilikom promene konteksta). Registar *Rp* se može samo čitati, jer je ožičen tako da njegova vrednost predstavlja jedinstveni identifikator svakog pojedinačnog procesora u multiprocesorskom sistemu. Svi registri su 32-bitni, a adresibilna jedinica je bajt.

Za multiprocesorski sistem sa ovim procesorom pravi se operativni sistem. U kernelu tog sistema postoje sledeće definisane konstante i strukture podataka:

```
const int NumOfProcessors = ... // Number of processors
struct PCB;  // Process Control Block
PCB* runningProcesses[NumOfProcessors]; // Running processes
```

U strukturi PCB postoje polja za čuvanje vrednosti svih registara *R*0..*R*31, PSW i SP. Pomeraji ovih polja u odnosu na početak strukture PCB simbolički označavaju sa `offs_r0` itd. Niz `runningProcesses`, u svakom svom elementu *n*, sadrži pokazivač na PCB onog procesa koji se trenutno izvršava na procesoru broj *n* (*n*=0..*NumOfProcesses*-1). Na raspolaganju je operacija `schedule()` (bez argumenata), koja vrši odabir narednog procesa za izvršavanje na procesoru na kome se izvršava i koja upisuje adresu PCB tog procesa u odgovarajući element niza `runningProcesses`.

a)(7) Na asembleru datog procesora napisati operaciju `dispatch()` (bez argumenata) koja čuva kontekst tekućeg izvršavanja i restaurira kontekst izvršavanja procesa kome treba dati procesor. U asembleru datog procesora može se koristiti identifikator statički alociranog podatka iz C programa, pri čemu se takva upotreba prevodi u konstantu sa vrednošću adrese tog podatka. Ova operacija se izvršava u kodu kernela, u sistemskom režimu. U ovu operaciju ulazi se iz prekidne rutine koja obrađuje sistemski poziv, pa su prekidi već maskirani (ne treba ih maskirati i demaskirati).

b)(3) Da li je u ovu operaciju neophodno ubaciti kod za međusobno isključenje konkurentnog izvršavanja od strane različitih procesora tehnikom uposlenog čekanja (*spin lock*)? Obrazložiti.

Rešenje:

4. (10 poena)

Korišćenjem niti u školskom jezgru realizovati klasu `Search` koja ima dole dati interfejs. Objekat ove klase obavlja pretragu datog niza celih brojeva `a` u opsegu indeksa počev od `i` zaključno sa `j`, tražeći u njemu vrednost `x`, ali tako što pretragu obavlja konkurentno i binarno-rekurzivno: jedna već kreirana nit nastavlja pretragu jedne polovine datog niza, a kreira novu nit koja vrši istu takvu pretragu druge polovine niza, i tako dalje rekurzivno. Ako se na nekom elementu *n* pronađe vrednost *x*, treba ispisati „Found at *n*!“, a ako se tu ne nađe ta vrednost, treba ispisati „Not found at *n*!“.

```
class Search : public Thread {
public:
  Search (int a[], int i, int j, int x);
};
```

Primer korišćenja ove klase je sledeći:

```
int array[N];
Thread *t = new Search(array,0,N-1,5);
t->start();
```

Rešenje: