Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1 (IR2OS1)
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika
Kolokvijum: Prvi, april 2013.
Datum: 27.4.2013.

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 tri uređaja; prva dva uređaja su ulazni, a treći je 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 program koji učitava po jednu reč sa bilo kog od dva ulazna uređaja (na kom god se pre pojavi spreman podatak) i odmah tu učitanu reč šalje na izlazni uređaj. 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)

U nekom sistemu sa straničnom organizacijom memorije virtuelni adresni prostor je veličine 4 GB, adresibilna jedinica je 32-bitna reč, a fizička adresa je veličine 32 bita. Ceo virtuelni adresni prostor ima 64 K stranica.

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

b)(5) Koliko 32-bitnih reči zauzima PMT? Obrazložiti.

Rešenje:

a)

b) Odgovor: _____

Obrazloženje:

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. Prilikom obrade sistemskog poziva `sys_call` treba uraditi redom sledeće:

1. Sačuvati kontekst tekućeg korisničkog procesa u njegov PCB. Procesor je RISC, sa *load/store* arhitekturom i ima sledeće programski dostupne registre: 32 registra opšte namene (R0..R31), SP, PSW i PC. Prilikom obrade prekida, procesor prelazi u sistemski (zaštićeni, kernel) režim, a na steku čuva samo PSW i PC. Na PCB tekućeg korisničkog procesa ukazuje globalni pokazivač koji se nalazi na adresi simbolički označenoj sa `running`. Ostale registre (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 `offsSP`, `offsR0`, ..., `offsR31`.

2. Preći na stek kernela u kome se izvršava ceo kernel kod. Na vrh tog steka (zapravo dno, pošto je stek prazan inicijalno, kao i posle svakog izlaska iz kernel koda) ukazuje pokazivač na lokaciji simbolički označenoj sa `kernelStack`.

3. Dozvoliti spoljašnje maskirajuće prekide instrukcijom `inte`, pošto je kod kernela i svih prekidnih rutina napravljen tako da se pojava prekida samo pamti u softverskim indikatorima, a kernel kod obrađuje prekid odloženo, kada je to bezbedno i odgovarajuće.

4. Pozvati potprogram na adresi simbolički označenoj sa `kernel`. Ovaj potprogram (bez argumenata) obavlja sve potrebne poslove, kao što su obrada sistemskog poziva i raspoređivanje (nakon čega `running` pokazuje na PCB novog tekućeg procesa).

5. Po povratku iz potprograma `kernel`, maskirati prekide instrukcijom `intd`, povratiti kontekst novog tekućeg procesa iz njegovog PCB-a, i vratiti se u korisnički režim i proces iz sistemskog poziva. Instrukcija `iret` prebacuje procesor u korisnički režim i vraća sačuvane registre PC i PSW sa tekućeg steka.

Na asembleru datog procesora napisati prekidnu rutinu `sys_call`.

Rešenje:

4. (10 poena)

U nekom operativnom sistemu postoji sledeći sistemski poziv:

```
int thread_create(void (*)(void*), void*);
```

koji kreira nit nad funkcijom na koju ukazuje prvi argument. Ta funkcija prima jedan netipizirani pokazivač i ne vraća rezultat. Novokreirana nit poziva tu funkciju sa stvarnim argumentom jednakim drugom argumentu ovog sistemskog poziva. Sistemski poziv vraća PID kreirane niti.

Korišćenjem ovog sistemskog poziva realizovati klasu `Thread` poput one u školskom jezgru, sa sledećim interfejsom:

```
class Thread {
public:
  void start ();
  virtual ~Thread ();
protected:
  Thread ();
  virtual void run () {}
};
```

Destruktor ove klase treba da sačeka da se nit predstavljena ovim objektom završi. U tu svrhu postoji sistemski poziv `wait(int pid)` koji suspenduje pozivajuću nit dok se nit sa datim PID ne završi.

Rešenje: