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, maj 2012.
Datum: 5.5.2012.

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 uređaja:

```
typedef unsigned int REG;
REG* dmaCtrl =...; // control register
REG* dmaStatus =...; // status register
REG* dmaBlkAddr =...; // data block address register
REG* dmaBlkSize =...; // data block size register
```

U upravljačkom registru najniži bit je bit *Start* kojim se pokreće prenos preko DMA, a u statusnom registru najniži bit je bit spremnosti (*Ready*) čija vrednost 1 ukazuje da je DMA spreman za novi prenos podatka (inicijalno je tako postavljen). Postavljanje bita spremnosti kada DMA završi zadati prenos generiše signal zahteva za prekid procesoru.

Zahtevi za izlaznim operacijama na nekom izlaznom uređaju vezani su u jednostruko ulančanu listu. Zahtev ima sledeću strukturu:

```
struct OutputRequest {
  char* buffer; // Buffer with data (block)
  unsigned int size; // Buffer (blok) size
  void (*callBack)(OutputRequest*); // Call-back function
  OutputRequest* next; // Next in the list
};
```

Kada se završi prenos zadat jednim zahtevom, potrebno je pozvati funkciju na koju ukazuje pokazivač `callBack` u tom zahtevu, sa argumentom koji ukazuje na taj zahtev. Ovu funkciju implementira onaj ko je zahtev postavio i služi da mu signalizira da je zahtev obrađen. Obrađeni zahtev ne treba brisati iz liste (to je odgovornost onog ko je zahtev postavio).

Potrebno je napisati kod operacije `transfer()`, zajedno sa odgovarajućom prekidnom rutinom `dmaInterrupt()`, koja obavlja sve prenose zadate zahtevima u listi na čiji prvi zapis ukazuje argument `ioHead`.

```
void transfer (OutputRequest* ioHead);
interrupt void dmaInterrupt ();
```

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 24h, stranici broj 45h u tom segementu, reč broj 15h u toj stranici, ako se ta stranica preslikava u okvir broj FF00h.

3. (10 poena)

Korišćenjem standardnih bibliotečnih funkcija `setjmp()` i `longjmp()`, realizovati operaciju

```
yield(jmp_buf old, jmp_buf new);
```

koja čuva čuva kontekst niti čiji je `jmp_buf` dat kao prvi argument, oduzima joj procesor i restaurira kontekst niti čiji je `jmp_buf` dat kao drugi argument, kojoj predaje procesor.

Koristeći ovu operaciju `yield()`, realizovati operaciju `dispatch()` koja ima isti efekat kao i ona data u školskom jezgru.

Rešenje:

4. (10 poena)

Klasa `Node`, čija je delimična definicija data dole, predstavlja čvor binarnog stabla. Funkcije `getLeftChild()` i `getRightChild()` vraćaju levo, odnosno desno podstablo datog čvora (tačnije, njegov levi i desni čvor-potomak).

```
class Node {
public:
  Node* getLeftChild();
  Node* getRightChild();
  ...
};
```

Potrebno je prebrojati čvorove u stablu rekurzivnim obilaskom datog binarnog stabla korišćenjem uporednih niti na sledeći način. Ako neka nit trenutno obilazi neki čvor, onda ona treba da napravi novu nit-potomka koja će obići desno podstablo tog čvora, a sama ta nit će nastaviti sa obilaskom levog podstabla tog čvora, i tako rekurzivno.

Realizovati globalnu operaciju `size(Node* root)` koja prebrojava čvorove i vraća broj čvorova u stablu sa datim korenim čvorom na opisani način. Niti treba kreirati sistemskim pozivom `fork()` za koga treba pretpostaviti da ima istu sintaksu i značenje povratne vrednosti kao i istoimeni Unix sistemski poziv, samo što umesto procesa kreira nit u istom adresnom prostoru roditelja: novokreirana nit ima istu poziciju u izvršavanju kao i roditeljska nit, deli isti adresni prostor, samo što poseduje sopstveni kontrolni stek koji inicijalno predstavlja identičnu kopiju roditeljskog steka u trenutku poziva `fork()`. Sistemski poziv `wait(null)` suspenduje pozivajuću nit sve dok se ne završe sve niti-potomci, a poziv `exit()` gasi pozivajuću nit.

Rešenje: