Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, mart 2018.
Datum: 24. 3. 2018.

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

Ukupno: _____/30 = _____% = _____/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 ulazna uređaja:

```
typedef unsigned REG;
REG* io1Ctrl =...;   // Device 1 control register
REG* io1Data =...;   // Device 1 data register
REG* io1Stat =...;   // Device 1 status register
REG* io2Ctrl =...;   // Device 2 control register
REG* io2Data =...;   // Device 2 data register
REG* io2Stat =...;   // Device 2 status register
REG* timer =...;     // Timer
```

U upravljačkim registrima bit 0 je bit *Start*. Spremnost novog ulaznog podatka prvi uređaj ne signalizira nikakvim signalom, ali je sigurno da je on spreman najkasnije 50 ms nakon preuzimanja prethodnog podatka (čitanja podatka iz registra), odnosno od postavljanja bita *Start*, ako se radi o prvom podatku. Spremnost novog ulaznog podatka drugi uređaj signalizira bitom spremnosti u razredu 0 svog statusnog registra, čije postavljanje ne generiše prekid procesoru.

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

Potrebno je napisati kod, uključujući i prekidnu rutinu od vremenskog brojača, koji vrši prenos po jednog bloka podataka sa svakog od dva uređaja uporedo. Prenos se obavlja pozivom sledeće funkcije iz koje se vraća kada su oba prenosa završena:

```
void transfer (unsigned* blk1, int count1, unsigned* blk2, int count2);
```

Rešenje:

2. (10 poena)

U implementaciji jezgra nekog jednoprocesorskog *time-sharing* operativnog sistema, radi pojednostavljenja celog mehanizma promene konteksta, primenjeno je sledeće neobično rešenje. Promena konteksta vrši se isključivo kao posledica prekida, na samo jednom mestu u kodu koji se izvršava na prekid. Prekidi dolaze od raznih uređaja, u najmanju ruku od vremenskog brojača, jer on u svakom slučaju generiše prekid zbog toga što je tekućoj niti isteklo dodeljeno procesorsko vreme. Zbog toga tekuća nit nikada ne gubi procesor sinhrono, čak ni kada poziva blokirajuću operaciju (sistemski poziv). Umesto toga, ukoliko je potrebno da se nit suspenduje (blokira) u nekom blokirajućem pozivu, nit se samo „označi" suspendovanom i nastavlja sa izvršavanjem (uposlenim čekanjem) sve dok ne stigne sledeći prekid. Kada takav prekid stigne, prekidna rutina vrši samu promenu konteksta.
Na primer, implementacija operacije `suspend`, koja suspenduje pozivajući proces, i `resume`, koja ponovo deblokira dati proces, izgledaju ovako:

```
void suspend () {
  running->status = suspended; // Mark as suspended
  while (running->status==suspended); // and then busy-wait
  // until it is preempted, suspended, and then resumed later
}

void resume (int pid) {
  processes[pid].status = ready; // Mark as ready
}
```

Procesor je RISC sa *load*-store arhitekturom, ima 32 registra opšte namene i SP. Prilikom prekida na steku čuva samo PC i PSW. Tajmer se restartuje upisom odgovarajuće vrednosti u registar koji se nalazi na adresi simbolički označenoj sa `Timer`. PCB procesa je dat strukturom definisanom dole, a svi procesi zapisani su u nizu `processes`. U asembleru, simboličko ime polja strukture ima vrednost pomeraja (engl. *offset*) tog polja od početka strukture.

```
enum ProcessStatus { unused, initiating, terminating, ready, suspended };
typedef unsigned long Time;
typedef unsigned Register;

struct PCB {
  ProcessStatus status;  // Process status
  Time timeSlice;        // Time slice for time sharing
  Register savedSP;      // Saved stack pointer
  ...
}

const unsigned long NumOfProcesses = ...;
PCB processes[NumOfProcesses];
PCB* running;  // Running process
```

a)(5) Na asembleru datog procesora napisati kod prekidne rutine koja vrši promenu konteksta. Ova prekidna rutina, pored čuvanja i restauracije konteksta na steku procesa, treba da pozove potprogram `scheduler` koji će u pokazivač `running` smestiti vrednost koja ukazuje na novoizabrani tekući proces, i da tajmer restartuje sa vremenskim kvantumom dodeljenim tom procesu. Pretpostavlja se da uvek postoji barem jedan spreman proces.

b)(5) Na jeziku C napisati potprogram `scheduler` koji bira sledeći proces za izvršavanje. Spremne procese treba da bira redom, u krug, a ne svaki put od početka niza `processes`.

Rešenje:

3. (10 poena)

Korišćenjem sistemskih poziva `fork`, `execlp` i `wait`, napisati program koji implementira jedan krajnje jednostavan interpreter komandne linije (engl. *command line interpreter*). Ovaj interpreter treba da učitava niz stringova razdvojenih belinama (razmacima ili novim redovima) sa standardnog ulaza, sve dok ne učita string „q“ koji prekida njegov rad. Kada učita svaki string, interpreter treba da pokrene proces nad programom zadatim stazom u tom stringu, sačeka njegov završetak, i pređe na sledeći string. Ukoliko kreiranje procesa nije uspelo, treba da ispiše poruku o grešci i pređe na sledeći. Pretpostavlja se da svaki string ima najviše 32 znaka. Podsetnik na bibliotečne funkcije i sistemske pozive koje se mogu koristiti:

- `scanf`: učitava sa standardnog ulaza; ukoliko se u formatizacionom specifikatoru, iza znaka %, napiše ceo broj, on označava maksimalan broj znakova koji će se učitati sa stadardnog ulaza; na primer, `%32s` učitava string, ali ne duži od 32 znaka (i dodaje `'\0'` na kraj);
- `void wait(int pid)`: suspenduje pozivajući roditeljski proces dok se ne završi proces-dete sa zadatim PID; ako je vrednost argumenta 0, pozivajući proces se suspenduje dok se ne završe sve procesi-deca;
- `int strcmp(char*,char*)`: poredi dva data stringa i vraća 0 ako su jednaki.

Rešenje: