Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika, Softversko inženjerstvo
Kolokvijum: Prvi, avgust 2023.
Datum: 27. 8. 2023.

Prvi kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. 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)

Na asembleru 32-bitnog procesora picoRISC napisati prevod dole date rekurzivne C funkcije `f`. Pretpostavke:

- vraćanje vrednosti tipa strukture iz funkcije implementira se tako što se u funkciju u registru R0 prenosi adresa lokacije u koju treba upisati povratnu vrednosti (u R0 je pokazivač na datu strukturu u koju pozivalac prihvata vraćenu vrednost);
- stek raste ka nižim adresama, a SP ukazuje na poslednju popunjenu lokaciju;
- prilikom poziva potprograma instrukcijom `call` procesor na stek najpre stavlja PSW, pa potom PC (svaki je 32-bitni); adresibilna jedinica je bajt.

```
struct S { int a, b; }

struct S f (int n) {
  struct S s;
  if (n==0) s.a = s.b = 1;
  else s = f(n-1);
  return s;
}
```

Rešenje:

2. (10 poena)

Neki sistem primenjuje kontinualnu alokaciju memorije za procese, s tim da tu memoriju alocira uvek u celim blokovima veličine `BLK_SIZE`, kako bi ublažio eksternu fragmentaciju. Adrese su 32-bitne (tip `uint32`). U strukturi PCB polje `baseBlk` označava broj memorijskog bloka koji je prvi alociran za proces (proces zauzima uvek memoriju počev od pomeraja 0 tog bloka), a polje `numOfBlocks` označava broj blokova memorije koje proces zauzima; proces uvek zauzima cele alocirane blokove i poravnat je na početak bloka. Evidenciju o slobodnim blokovima sistem vodi u bit-vektoru `freeMemBlks`; svaki bit ovog vektora predstavlja jedan blok memorije (redom od nižih prema višim elementima niza i redom od najnižeg do najvišeg bita svakog 32-bitnog elementa niza, blokovi su numerisani počev od 0), a vrednost 1 označava da je blok slobodan.

a)(2) Implementirati funkciju `getMemCtxt` koja treba da izračuna baznu adresu (i upiše je u parametar `base`) i najvišu dozvoljenu virtuelnu adresu (i upiše je u parametar `limit`) za dati proces. Ovu funkciju sistem poziva prilikom promene konteksta da bi izračunate vrednosti upisao u odgovarajuće registre procesora kada datom procesu dodeljuje procesor.

b)(3) Implementirati funkciju `isMemBlkFree` koja za dati redni broj memorijskog bloka vraća informaciju o tome da li je dati blok slobodan ili ne, kao i funkciju `allocMemBlk` koja memorijski blok sa zadatim brojem označava zauzetim.

c)(5) Implementirati funkciju `expand` koju sistem poziva kada tekući proces izazove izuzetak zbog prekoračenja najviše dozvoljene virtuelne adrese. Ova funkcija treba da pokuša da dodeli još jedan dodatan memorijski blok datom procesu, ako takav postoji iza prostora koji proces već zauzima, proširujući tako memorijski prostor procesa; u tom slučaju ova funkcija treba da vrati 0. U suprotnom, ova funkcija treba da vrati -1. Najviši blok fizičke memorije je uvek zauzet od strane jezgra operativnog sistema i nikad se ne dodeluje procesima.

```
extern uint32 freeMemBlks[];
extern size_t BLK_SIZE;

void getMemCtxt (PCB* pcb, uint32& base, uint32& limit);

bool isMemBlkFree (size_t num);
void allocMemBlk (size_t num);

int expand (PCB* pcb);
```

Rešenje:

3. (10 poena)

Neki sistem sa straničnom orgnizacijom virtuelne memorije obrađuje događaje vezane za alokaciju i zamenu stranica polimorfno, pozivom odgovarajućih funkcija dinamičkim vezivanjem preko pokazivača koji se nalaze u poljima odgovarajućih struktura. Konkretnije, u sledećim strukturama postoje sledeća polja koja su pokazivači na funkcije za obradu sledećeg:

- U strukturi `PCB` polje `handleMemSegFault` je pokazivačkog tipa `void (*)(PCB* pcb, uint32 page)` i pokazuje na funkciju koju treba pozvati ako proces sa datim PCB adresira datu stranicu broj `page` koja ne pripada nijednom alociranom logičkom segmentu virtuelne memorije; ova funkcija će dalje ugasiti proces ili uraditi nešto drugo ako je redefinisana.
- U strukturi `SegDsc`, koja predstavlja deskriptor jednog logičkog segmenta, polja `coldLoad` i `hotLoad` su istog pokazivačkog tipa `void (*)(SegDsc* sd, PCB* pcb, PageDsc* pd, uint32 page, void* frame)` i pokazuju na funkcije koje treba pozvati kada datu stranicu broj `page` datog procesa `pcb` kojoj odgovara deskriptor logičkog segmenta `sd` i deskriptor stranice `pd` treba učitati odnosno inicijalizovati u dati okvir `frame` kad se ta stranica koristi i učitava prvi (`coldLoad`) odnosno svaki naredni put (`hotLoad`).

U jednom bitu deskriptora stranice `PageDsc` u PMT sistem vodi evidenciju o tome da li je data stranica ranije već bila korišćena i alocirana ili nije. Za održavanje ovog bita na raspolaganju su sledeće funkcije:

- `bool isFirstAccess (PageDsc*)`: vraća `true` ako je u datom deskriptoru stranice stranica označena tako da joj se ranije nije pristupalo (pa zato prvo obraćanje zahteva učitavanje sa `coldLoad`);
- `void setAccessed (PageDsc*)`: u datom deskriptoru stranice označava stranicu kao ranije već korišćenu (pa zato svako naredno učitavanje zahteva poziv `hotLoad`).

Na raspolaganju su i sledeće funkcije:

- `void* allocFrame ()`: alocira jedan slobodan okvir memorije i vraća njegovu adresu. Ova funkcija brine o zameni stranica ukoliko slobodnog okvira nema (izabira stranicu-žrtvu za izbacivanje, po potrebi je snima), tako da uvek vraća validnu nenultu vrednost;
- `SegDsc* getSegDsc (PCB*, uint32 page)`: vraća pokazivač na deskriptor logičkog segmenta za dati proces i datu stranicu; ukoliko data stranica ne pripada nijednom alociranom logičkom segmentu, vraća *null*;
- `PageDsc* getPageDesc (PCB* pcb, uint32 page)`: vraća deskriptor date stranice u PMT datog procesa. Ova funkcija uvek vraća validnu nenultu vrednost, čak i ako je PMT organizovana u više nivoa, jer po potrebi alocira PMT višeg nivoa za traženu stranicu.

Implementirati sledeću funkciju:

```
void handlePageFault (PCB* pcb, uint32 page);
```

koju kernel poziva kada obrađuje hardverski izuzetak zbog stranične greške koju je izazvao dati proces adresiranjem date stranice. (Ovde se radi samo o straničnoj greški, ne o nedozvoljenoj operaciji.)

R
e
š
e
n
j