Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, april 2026.
Datum: 23. 4. 2026.

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)

Dat je deo nekog programa na asembleru za picoRISC koji neki proces izvršava. Ovaj deo programa kopira sadržaj celobrojnog niza `b` u celobrojni niz `a`. Oba niza su alocirana statički u odvojene logičke segmente i počinju od virtuelnih adresa datih dole. Adresibilna jedinica je bajt, procesor je 32-bitni, tip `int` je takođe 32-bitni, a instrukcija uvek ima jednu 32-bitnu reč sa specifikacijom koda operacije i načina adresiranja, dok se eventualna konstanta koja predstavlja neposredni operand, pomeraj ili apsolutnu adresu uvek nalazi u drugoj 32-bitnoj reči instrukcije.

```
copyab: xor   r1,r1,r1
        load  r2,#4
        load  r3,#0x100
loop1:  load  r0,[b+r1]
        store r0,[a+r1]
        add   r1,r1,r2
        dec   r3
        jnz   loop1
```

a)(3) Ako su dati deo koda i statički nizovi `a` i `b` smešteni počev od datih 32-bitnih virtuelnih adresa (najniže adrese), navesti koje su završne virtuelne adrese (poslednjeg zauzetog bajta, najviša adresa) ovih elemenata kojima proces pristupa tokom izvršavanja. Vrednosti pisati heksadecimalno.

Deo virtuelnog adresnog prostora	Početna adresa (hex)	Završna adresa (hex)
Instrukcije (`copyab`)	`20EFFFFC`	
Niz `a`	`34CFFFF0`	
Niz `b`	`12000FF0`	

b)(3) Ako procesor i operativni sistem koriste kontinualnu organizaciju memorije, a posmatrani proces je smešten počev od bazne fizičke adrese 30000000h, navesti 32-bitne fizičke adrese koje dati elementi zauzimaju u fizičkoj memoriji tokom izvršavanja ovog procesa.

Deo virtuelnog adresnog prostora	Početna adresa (hex)	Završna adresa (hex)
Instrukcije (`copyab`)		
Niz `a`		
Niz `b`		

c)(3) Ako procesor i operativni sistem koriste segmentnu organizaciju memorije, navesti 32-bitne fizičke adrese koje dati elementi zauzimaju u fizičkoj memoriji tokom izvršavanja ovog procesa. Maksimalna veličina segmenta je 16 MB. Fizički segment posmatranog procesa u kome su smeštene date instrukcije je smešten počev od bazne fizičke adrese 30000000h, fizički segment u koji je smešten niz `a` je smešten of bazne adrese 40000000h, a fizički segment sa nizom `b` je smešten od bazne adrese 20000000h.

Deo virtuelnog adresnog prostora	Početna adresa (hex)	Završna adresa (hex)
Instrukcije (`copyab`)		
Niz `a`		
Niz `b`		

2. (10 poena)

Neki sistem sa straničnom organizacijom memorije za svaki proces organizuje evidenciju alociranih logičkih segmenata, koje naziva *regionima*, ulančanom listom struktura tipa `RegionDesc` koji je dat dole. Za svaki alocirani region ova struktura u polju `startPage` čuva broj početne stranice regiona, a polje `size` veličinu regiona izraženu u stranicama. Ulančana lista je uređena po rastućem redosledu vrednosti polja `startPage`. Broj poslednje stranice u virtuelnom adresnom prostoru koji je dostupan za regione koje alocira proces definisan je konstantom `MAX_PAGE_NUM`, a tip `size_t` je neoznačen celobrojni tip koji je bar nekoliko bita širi od brojeva stranica.

Ovaj sistem nudi procesima sistemski poziv kojim proces zahteva povećanje (proširenje prema višim adresama) za datu veličinu svog već alociranog regiona, ukoliko iza datog alociranog regiona ima dovoljno mesta za traženo proširenje, tj. ukoliko to proširenje ne bi prekoračilo naredni alocirani region ili kraj dostupnog virtuelnog adresnog prostora.

Implementirati funkciju `extendRegion` koja realizuje ovaj sistemski poziv za ulančanu listu deskriptora regiona čija je glava data kao prvi parametar. Parametar `startPage` definiše početnu stranicu regiona koji se proširuje, a parametar `by` veličinu za koju se proširuje izraženu u stranicama. Vrednost prvog parametra mora da se poklapa sa početkom nekog alociranog regiona. Ova funkcija treba da vrati status tipa `SCStatus` i to `ok` ako je usluga uspešno završena, `illegalArg` ako je bilo koji parametar nekorektan, odnosno `noMemory` ako za traženo proširenje nema dovoljno mesta.

```
enum SCStatus { ok, ..., illegalArg, ..., noMemory, ...};
const size_t MAX_PAGE_NUM = ...;

struct RegionDesc {
  size_t startPage, size;
  RegionDesc* next;
};

SCStatus extendRegion (RegionDesc* head, size_t startPage, size_t by);
```

Rešenje:

3. (10 poena)

Neki operativni sistem sa straničnom organizacijom virtuelne memorije i zamenom stranica logičke segmente virtuelne memorije predstavlja objektima klasa izvedenih iz apstraktne klase `Segment` sa polimorfnom operacijom `loadPage` koja se poziva kada sistem obrađuje straničnu grešku procesa na kog ukazuje prvi parametar (objekat klase `Process`), za adresiranu stranicu koja je data kao drugi parametar, a koju treba smestiti u okvir čiji je broj dat kao treći parametar (tipovi `Page` i `Frame` su sinonimu za neoznačene celobrojne tipove). Iz klase `Segment` izvedene su sledeće tri klase:

- `BSSSegment`: predstavlja *bss* segmente podataka i to onih statički inicijalizovanih nulama (podatak član `isZeroInit` ove klase je u tom slučaju postavljen na `true`), ili onih koji nisu inicijalizovani statički, nego će biti inicijalizovani dinamički (`isZeroInit` je `false`).
- `DataSegment`: predstavlja segment za podatke koji su inicijalizovani statički na proizvoljne vrednosti.
- `TextSegment`: predstavlja segment sa kodom programa (instrukcijama).

Implementirati funkcije `loadPage` ove tri klase. Ove funkcije treba da obezbede da je tražena stranica smeštena u dati okvir pri prvom ili svakom drugom obraćanju i vrati 0 u slučaju uspeha, odnosno kôd greške pri učitavanju stranice sa diska (ostale raspložive funkcije uvek vraćaju ispravan rezultat za ispravne parametre). Na raspolaganju je sledeće:

- Klasa `PageDesc` apstrahuje jedan ulaz za jednu stranicu u PMT datog procesa. Nestatička funkcija članica ove klase `isAccessed` vraća `true` akko je u ovom deskriptoru, u posebnom bitu koji održava operativni sistem, postavljena informacija o tome da je datoj stranici ranije već pristupano, a funkcija `setAccessed` ga postavlja.
- Klasa `Process` apstrahuje proces i čuva podatke o njemu u jezgru.
- Nestatička funkcija članica `getPageDesc` klase `Process` vraća pokazivač na deskriptor date stranice u PMT tog procesa.
- Nestatičke funkcije članice `loadExePage` i `loadDataPage` klase `Process` sa diska učitavaju datu stranicu u dati okvir i to iz *exe* fajla, odnosno iz prostora za zamenu, respektivno, i vraćaju status operacije učitavanja (0 u slučaju uspeha, inače kôd greške).
- Globalna funkcija `writeZeros` upisuje sve nule u dati okvir fizičke memorije.

```
int BSSSegment::loadPage (Process* pr, Page pg, Frame fr);
int DataSegment::loadPage (Process* pr, Page pg, Frame fr);
int TextSegment::loadPage (Process* pr, Page pg, Frame fr);

bool PageDesc::isAccessed () const;
void PageDesc::setAccessed ();
PageDesc* Process:getPageDesc (Page pg) const;
int Process::loadExePage(Page pg, Frame fr);
int Process::loadDataPage(Page pg, Frame fr);
void writeZeros (Frame fr);
```

R
e
š
e
n
j