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 2024.
Datum: 20. 3. 2024.

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 sadržaj dva asemblerska fajla (`a.asm` i `b.asm`) za 32-bitni procesor picoRISC. Direktivom `export` asembleru se nalaže da navedene simbole (definisane dalje u tom fajlu) „izveze“ kao simbole dostupne za spoljno vezivanje, dok se direktivom `import` „uvoze“ navedeni simboli definisani u nekom drugom fajlu (ovom direktivom ti simboli su deklarisani u datom fajlu, ali sa nepoznatom vrednošću). Nakon prevođenja (asembliranja), linkeru se zadaje da spoji dva fajla dobijena prevođenjem fajlova `a.asm` i `b.asm`, tim redom. Linker slaže segmente u virtuelni adresni prostor redom kojim na njih nailazi počev od virtuelne adrese 0, s tim da svaki nov segment počinje od početka nove stranice veličine 4 KB (adresibilna jedinica je bajt).

a.asm

```
export main, a
import max

seg data
    a dd 1, 2, 3, 4, 5
endseg

seg text
    main:   load r1,a
            push r1
            load r1,a+4
            push r1
            call max
            pop  r1
            pop  r1
            ret
endseg
```

b.asm

```
export max

seg text
    max:    load r1,[sp+8]
            load r2,[sp+12]
            cmp  r1,r2
            jle  L0001
            load r0,r1
            ret
    L0001:  load r0,r2
            ret
endseg
```

a)(3) Koje virtuelne adrese dodeljuje linker simbolima `main`, `a` i `max` tokom povezivanja (napisati ih heksadecimalno)? Obrazložiti odgovor.

b)(3) Ukoliko instrukcija `call` na vrh steka stavlja samo PC, koje tri 32-bitne vrednosti se nalaze na vrhu steka prilikom ulaska u potprogram `max` pri izvršavanju procesa pokrenutog nad navedenim povezanim programom? Vrednosti navesti heksadecimalno u poretku kojim su stavljene na stek – poslednju onu na vrhu steka.

c)(4) Ukoliko se primenjuje stranična organizacija memorije, kolika je interna fragmentacija za sadržaj ova dva fajla (koliko je neiskorišćene memorije koja je alocirana za logičke segmente u ovim fajlovima)? Obrazložiti odgovor.

Rešenje:

2. (10 poena)

Neki sistem primenjuje kontinualnu alokaciju memorije, ali u jedinicama koje su blokovi memorije određene konstantne veličine (alokacija se uvek radi u delovima veličine zaokružene na cele blokove i poravnato na blokove). Za evidenciju slobodnih i zauzetih blokova memorije sistem koristi posebnu strukturu zapisanu u nizu `memMap` veličine `NumOfBlocks` (ukupan broj blokova dostupnih za alokaciju). Svaki element ovog niza odgovara jednom bloku memorije koja se evidentira ovim nizom: elementi ovog niza redom, jedan na jedan odgovaraju blokovima memorije koji se alociraju. Za određeni slobodan fragment memorije koji počinje u bloku sa rednim brojem `b` i koji je veličine `k` susednih blokova, element `memMap[b]` ima vrednost `k`, dok narednih `k-1` elemenata ovog niza ima vrednost 0. Slično, za zauzet segment memorije koji počinje u bloku sa rednim brojem `b` i ima veličinu `k` susednih blokova, element `memMap[b]` ima vrednost `-k` (negativna vrednost), dok narednih `k-1` elemenata ima vrednost 0. Inicijalno je element sa indeksom 0 ovog niza jednak `NumOfBlocks`, dok su svi ostali elementi niza jednaki 0 (cela memorija je slobodna).

Implementirati funkciju `alloc` koja treba da alocira deo memorije zadate veličine `size` susednih blokova po *first fit* algoritmu i vrati redni broj prvog bloka koji je alociran, a u slučaju da slobodnog prostora nema, vraća -1.

```
const int NumOfBlocks = ...;
int memMap[NumOfBlocks] = {NumOfBlocks};
int alloc (int size);
```

Rešenje:

3. (10 poena)

Neki sistem sa straničnom organizacijom memorije i alokacijom stranica na zahtev (*demand paging*) omogućava logičko deljenje (logičkih) segmenata virtuelne memorije, ali tako da najviše dva procesa mogu da dele jedan segment. U svakom od tih procesa deljeni logički segment može da zauzima različite pozicije u virtuelnom adresnom prostoru (ali iste veličine). Segment jednog procesa opisan je objektom klase `SegDesc`. PMT procesa kom pripada dati segment dat je u polju `pmt`, dok je broj prve stranice tog segmenta dat u polju `startPage`. Ukoliko je segment deljen, segment drugog procesa sa kojim je dati segment deljen dat je u polju `sharedSeg` (ima vrednost *null* ako segment nije deljen).

Data polimorfna operacija `SegDesc::initPage` po potrebi učitava, odnosno inicijalizuje datu stranicu u dati okvir prethodno alociran za tu stranicu (ona ne menja PMT), u zavisnosti od vrste segmenta; ova funkcija vraća status izvršenja (0 za uspeh). PMT je predstavljen klasom `PMT`, čija funkcija `getFrame` vraća broj okvira za datu stranicu (0 ako stranica nije alocirana), a funkcija `setFrame` postavlja broj okvira za datu stranicu u PMT.

Implementirati funkciju `SegDesc::allocPage` koja se poziva prilikom obrade stranične greške i koja treba da obezbedi da data stranica za dati PMT procesa koji je izazvao straničnu grešku bude alocirana, učitana u memoriju i postavljen njen ulaz u PMT; uzeti u obzir slučaj kada je stranica deljena i već alocirana od strane procesa sa kojim je deljena. Provera da data stranica pripada datom segmentu je već obavljena. Pretpostavlja se da su polja za prava pristupa u PMT uvek inicijalizovana za sve stranice alociranih segmenata, čak i ako stranice nisu alocirane. Ova operacija treba da vraća status izvršenja (0 za uspeh).

```
typedef Frame uint16;
typedef Page uint32;
Frame getFreeFrame ();

class SegDesc {
public:
  int allocPage (Page page);
  ...
protected:
  virtual int initPage (Page, Frame);
  ...
private:
  Page startPage;
  PMT* pmt;
  SegDesc* sharedSeg = nullptr;
  ...
};

class PMT {
public:
  Frame getFrame (Page) const;
  void  setFrame (Page, Frame);
  ...
};
```

Rešenje: