Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika
Kolokvijum: Prvi, april 2024.
Datum: 28. 4. 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 jednog izvornog fajla sa C kodom.

```
int a[5] = {1, 2, 3, 4, 5}, n = 0;

int max (int a[], int n) {
  if (n<=1) return a[0];
  int m = max(a+1,n-1);
  return (a[0]>m) ? a[0] : m;
}
```

a)(6) Napisati asemblerski kod za 32-bitni procesor picoRISC, sa sintaksom direktiva pokazanim na predavanjima, kakav bi prevodilac mogao da napravi prevođenjem ovog fajla. Logički segment se na asembleru deklariše direktivom `seg` uz koju ide kvalifikator za tip segmenta (`text`, `bss` ili `data`); npr. `seg text`. Stek raste ka nižim adresama, SP ukazuje na prvu slobodnu lokaciju, adresibilna jedinica je bajt, a instrukcija poziva potprograma na steku čuva PC i PSW tim redom.

b)(2) Ukoliko asembler koristi relativno adresiranje (u odnosu na PC) kod prevođenja instrukcije `call` kad god može, koja vrednost će biti generisana u drugoj 32-bitnoj reči mašinske instrukcije `call` koja radi rekurzivan poziv (napisati heksadecimalno)?

c)(2) Da li se navedeno relativno adresiranje za instrukciju `call` može koristiti i za poziv funkcije `max` iz drugog logičkog segmenta koji se nalazi u drugom izvornom asemblerskom fajlu? Kratko obrazložiti odgovor.

Rešenje:

2. (10 poena)

Neki sistem sa straničnom organizacijom virtuelne memorije i dohvatanjem stranica na zahtev (*demand paging*) ima PMT u dva nivoa, s tim da operativni sistem i PMT drugog nivoa takođe alocira na zahtev, tek kada proces prvi put ispravno pristupi nekoj stranici iz opsega te PMT drugog nivoa. Zato su inicijalno svi ulazi u PMT prvog nivoa *null*. Širina virtuelne adrese je 32 bita, stranica ima 4K adresibilnih jedinica, a PMT oba nivoa isti broj ulaza.

a)(7) Dole su date deklaracije tipova za stranice prvog i drugog nivoa. Implementirati operaciju `getPageDesc` koja za datu virtuelnu adresu vraća descriptor stranice u PMT kojoj ta virtuelna adresa pripada. U slučaju da PMT1 drugog nivoa nije alocirana, treba je alocirati. Za alokaciju memorije za svoje strukture kernel koristi svoju internu funkciju `kmalloc` koja je na raspolaganju. U slučaju da nema prostora za PMT1, funkcija `getPageDesc` treba da vrati *null*.

b)(3) Ukratko, ali precizno objasniti šta operativni sistem treba da uradi prilikom obrade stranične greške u opisanom sistemu.

```
const uint32 offsetw = 12;
const uint32 page1w = 10;
const uint32 PMT0_size = 1024;
const uint32 PMT1_size = 1024;

typedef uint32 PageDesc;
typedef PageDesc PMT1[PMT1_SIZE];
typedef PMT1* PMT0[PMT0_SIZE];
void* kmalloc (size_t size);

inline PageDesc* getPageDesc (PMT0 pmt, void* vaddr);
```

Rešenje:

3. (10 poena)

U nekom fajlu zapisana su dva veoma velika celobrojna niza iste zadate veličine `arr_size`. Najpre je, počev od bajta 0, zapisano `arr_size` elemenata niza `a` tipa `int`, a odmah iza toga isto toliko elemenata niza `b`. Iza toga je u fajlu obezbeđen prostor za smeštanje istog tolikog niza `c`. Celobrojni elementi su veličine 4 bajta, a u fajlu su zapisani u istom formatu u kom se i smeštaju u operativnu memoriju (niži bajt na nižoj adresi, *little endian*).

Potrebno je realizovati funkciju `arr_add` koja treba da sabere ova dva niza `a` i `b` (element po element) i njihov rezultat zapiše na mesto obezbeđeno za niz `c` u datom fajlu. Ona se realizuje za neki skroman mikroračunar za vrlo malo RAM-a, tako da za smeštanje (delova) nizova u memoriju ukupno ne treba utrošiti više od 32 KB RAM-a. Na raspolaganju su funkcije `read_block` i `write_block` koje iz fajla učitavaju, odnosno u fajl upisuju (respektivno) `size` celih brojeva smeštenih na adresu datu pokazivačem `buffer`, počev od bajta sa rednim brojem `offset` u navedenom fajlu.

```
void read_block (size_t offset, size_t size, int* buffer);
void write_block (size_t offset, size_t size, const int* buffer);
void arr_add (size_t arr_size);
```

Rešenje: