
Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Odsek: Računarska tehnika i informatika, Softversko inženjerstvo
Kolokvijum: Prvi, avgust 2026.
Datum: 6. 9. 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)

U nekom jezgru postoji funkcija `memcpy_usr2krnl` čiji je potpis dat dole. Ova funkcija koristi se u implementaciji različitih sistemskih poziva i zadatak joj je da iskopira sadržaj dela memorije u virtuelnom adresnom prostoru tekućeg korisničkog procesa – niz 32-bitnih reči veličine `sz` na koji ukazuje pokazivač `usrbuf` (sadrži virtuelnu adresu u memorijskom kontekstu tekućeg korisničkog procesa) u isti toliki deo na koji ukazuje pokazivač `krnlbuf` u memorijskom prostoru koji koristi jezgro.

```
void memcpy_usr2krnl(const uint32_t* usrbuf, size_t sz, uint32_t* krnlbuf);
```

Implementirati ovu funkciju za dva odvojena slučaja i na dva načina:

a)(5) Memorijski prostor koji koristi jezgro za svoj kôd i podatke preslikava se u isti deo virtuelnog adresnog prostora svakog procesa. Implementirati ovu funkciju na assembleru procesora picoRISC. Dati parametri se redom prosleđuju u registrima `r1`, `r2` i `r3`, registar `r0` je slobodan za korišćenje, kao i svi ostali registri opšte namene, jer je pozivalac dužan da pre poziva potprograma na steku sačuva sve registre čije su vrednosti potrebne i nakon poziva tog potprograma (i restaurira ih nakon povratka iz njega).

b)(5) Memorijski prostor koji koristi jezgro za svoj kôd i podatke nije deo virtuelnog adresnog prostora procesa, ali se tokom izvršavanja kôda jezgra sve generisane adrese smatraju fizičkim (nema preslikavanja adresa). Implementirati ovu funkciju na jeziku C. Data je funkcija jezgra:

```
int32_t* v2p_addr (PMT* pmt, uint32_t* ptr);
```

Ova funkcija vraća fizičku adresu u koju se preslikava data virtuelna adresa `ptr` u virtuelnom adresnom prostoru procesa na čiji PMT ukazuje prvi parametar. Na PMT tekućeg procesa za koji se poziva funkcija `memcpy_usr2krnl` ukazuje pokazivač dostupan kao `running->pmt`.

Rešenje:

2. (10 poena)

U nekom sistemu adresibilna jedinica je bajt, virtuelni adresni prostor je veličine 4 GB, a fizički adresni prostor je veličine 1 GB. Sistem koristi segmentno-straničnu organizaciju memorije sa stranicom veličine 1 KB i 64 fizička segmenta. Jedan ulaz u SMT zauzima dve 32-bitne reči; u nižoj je pokazivač (cela adresa) na PMT prvog nivoa (ili 0 ako segment nije alociran), a u višoj reči su prava pristupa (RWX) u najniža tri bita i maksimalan dozvoljen broj stranice u segmentu (*limit*) u bitima do njih; preostali biti se ne koriste. PMT za svaki segment je organizovana u dva nivoa. PMT oba nivoa imaju isti broj ulaza, a svaki ulaz je 32-bitni. Ulaz u PMT prvog nivoa sadrži fizičku adresu PMT drugog nivoa, a ulaz u PMT drugog nivoa u nižim bitima sadrži broj okvira u koji je smeštena stranica (ako jeste alocirana).

Da bi obradio sistemske pozive u kojima se neki parametar zadaje kao pokazivač (virtuelna adresa u adresnom prostoru pozivajućeg procesa), operativni sistem mora da konvertuje datu virtuelnu u fizičku adresu, jer se kod kernela izvršava u režimu bez preslikavanja adresa. Implementirati funkciju koja obavlja ovu konverziju (vraća *null* ako virtuelna adresa ne pripada alociranom delu virtuelnog adresnog prostora). Prvi parametar ove funkcije je pokazivač na SMT pozivajućeg procesa. U bajt na koji ukazuje treći parametar (ako on nije *null*) treba upisati pročitane bite prava pristupa iz SMT. Napisati sve potrebne deklaracije za tipove SMT i PMT.

```
void* v2pAddr (SMT pmt, void* vaddr, uint8_t* rwx);
```

Rešenje:

3. (10 poena)

Neki sistem sa straničnom organizacijom memorije koristi zamenu stranica, pri čemu izbačene stranice snima u prostor za zamenu (*swap space*). U deskriptoru stranice u PMT postoji bit prisutnosti stranice u memoriji koji koristi hardver: ako je vrednost ovog bita 0, hardver generiše straničnu grešku. Ako je stranica bila u memoriji, pa je izbačena iz nje i snimljena u prostor za zamenu, odakle je treba učitati kada je potrebno, u isto polje u kome se nalazi broj okvira kada stranica jeste u memoriji operativni sistem upisuje vrednost koju naziva *swapBlk* i koja ukazuje na mesto u prostoru za zamenu u koji je stranica upisana. Ako stranica nije u prostoru za zamenu, recimo zato što joj se do sada nije ni pristupalo ili je nema potrebe snimati u prostor za zamenu, ovo polje ima vrednost 0.

Realizovati funkciju `handlePF` koju OS izvršava kada obrađuje straničnu grešku za virtuelnu adresu koja je data kao argument. Ova funkcija je nestatička članica klase `MemContext` koja sadrži sve informacije o memorijskom kontekstu datog procesa. Ova funkcija vraća vrednost iz enumeracije `Status`, za koju treba pretpostaviti postojanje svih potrebnih enumeratora za različite vrste grešaka (npr. `ERR_INVALID_ADDR`, `ERR_NO_FREE_SPACE` itd.), kao i enumeratora `OK`.

```
Status MemContext::handlePF (void* va);
```

Na raspolaganju su sledeće funkcije:

- `Page MemContext::getPage(void* va):` vraća broj stranice (tipa `Page`) za datu virtuelnu adresu;
- `MemRegion* MemContext::getMemRegion(Page pg):` za datu stranicu u datom memorijskom kontekstu vraća pokazivač na objekat klase `MemRegion` koji predstavlja alociran logički segment (region) u virtuelnom adresnom prostoru procesa kom ta stranica pripada; ako data stranica ne pripada alociranom regionu, vraća *nullptr*;
- `void* getFreeFrame():` alocira slobodan okvir u operativnoj memoriji i vraća njegovu adresu, po potrebi izbacujući neku stranicu iz memorije;
- `PageDesc* MemContext::getPageDesc(Page pg):` u PMT datog memorijskog konteksta pronalazi deskriptor date stranice (ulaz u PMT za datu stranicu), po potrebi alocirajući PMT u više nivoa ako ne postoje; polje *swapBlk* tipa `SwapBlk` u ovom deskriptoru sadrži *swapBlk*;
- `Status MemRegion::loadPageFromSwap(SwapBlk sb, void* frame):` učitava sadržaj izbačene stranice iz datog bloka prostora za zamenu u dati okvir;
- `Status MemRegion::loadInitPage(Page pg, void* frame):` inicijalizuje sadržaj date stranice u dati okvir ili je učitava iz njenog izvorišta (*exe* fajla), u zavisnosti od vrste regiona.

Rešenje: