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, maj 2026.
Datum: 31. 5. 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)

Data je funkcija `vsum` na jeziku C koja rekurzivno izračunava zbir elemenata niza celih brojeva na koji ukazuje prvi argument. Veličina niza data je drugim argumentom.

```
static int a[0x100];

int vsum (const int* p, unsigned int n) {
  if (n>0) return *p + vsum(p+1,n-1);
  else return 0;
}
```

a)(5) Napisati prevod ove funkcije na asembler za procesor picoRISC uz poštovanje sledeće konvencije prenosa argumenata u funkcije. Prvi argument (sleva) prenosi se u registru R1, drugi u registru R2 itd. Ako je pozivaocu potrebna vrednost nekog od njegovih argumenata u navedenim registrima nakon poziva druge funkcije, mora ga sačuvati na steku pre poziva i restaurirati nakon povratka iz te druge funkcije. Povratna vrednost prenosi se kroz R0.

b)(5) Virtuelna adresa je 32-bitna, adresibilna jedinica je bajt, a memorija je organizaovana stranično sa stranicom veličine 4 KB. Ako su prevedena funkcija `vsum` i statički niz `a` smešteni počev od datih najnižih virtuelnih adresa, navesti najviše virtuelne adrese koje dati elementi zauzimaju u memoriji tokom izvršavanja ovog procesa (poslednja zauzeta lokacija), kao i brojeve stranica u kojima se oni nalaze. Ako stek raste ka nižim adresama, SP ukazuje na poslednju zauzetu lokaciju, a odmah po ulasku u funkciju pozvanu sa `vsum(a,0x100)` ima vrednost 0x12001000, navesti broj stranice ili stranica koje će stek zauzimati tokom izvršavanja do povratka iz ovog poziva (uključujući i taj povratak).

Deo virtuelnog adresnog prostora	Najniža adresa (hex)	Najviša adresa (hex)	Zauzete stranice
Instrukcije (`vsum`)	`20EFFFFC`		
Niz `a`	`14CFFFF0`		
Stek	`-`	`12001000`	

2. (10 poena)

Neki operativni sistem sa straničnom organizacijom memorije implementira zauzimanje i oslobađanje slobodnih okvira memorije klasom `FrameAllocator` datom dole. Evidencija slobodnih okvira vodi se u bit vektoru – članu `bits` ove klase koji je inicijalizovan svim nulama. Implementirati operacije `alloc` i `free` ove klase. Funkcija `alloc` treba da pronađe prvi slobodan okvir i zauzme ga, a vrati njegov broj; ako slobodnog okvira nema, treba da vrati -1. Funkcija `free` oslobađa okvir sa datim brojem. Na raspolaganju je funkcija `getLSBZero(uint32_t)` koja vraća poziciju najnižeg bita jednakog 0 u datoj 32-bitnoj reči i implementirana je odgovarajućom mašinskom instrukcijom.

```
class FrameAllocator {
public:
  size_t alloc ();
  void   free (size_t frame);
private:
  uint32_t bits[BV_SIZE] = {};
};
```

Rešenje:

3. (10 poena)

U nekom sistemu postoje sledeći sistemski pozivi vezani za deljene biblioteke sa dinamičkim vezivanjem (DLL):

- `int mapDLL (const char* dllName)`: po potrebi učitava DLL sa zadatim imenom i mapira ga u virtuelni adresni prostor pozivajućeg procesa; u slučaju uspeha, vraća pozitivnu celobrojnu vrednost koja predstavlja identifikator datog DLL-a u kontekstu procesa; u slučaju greške, vraća negativnu vrednost;
- `int mapDLLSymbol (int dll, const char* symbolName)`: u DLL-u koji je prethodno mapiran pronalazi simbol sa zadatim imenom i vraća njegovu (virtuelnu) adresu; u slučaju neuspeha, vraća *null*.

Neki program napravljen je tako da koristi dinamičko učitavanje svojih modula, s tim da za implementaciju modula i dinamičkog učitavanja koristi DLL-ove i opisane sistemske pozive. U DLL-u „mydll_1.dll“ definisana je funkcija `f1`, a u DLL-u „mydll_2.dll“ funkcija `f2`; potpisi ovih funkcija dati su dole. Simboli za njih kodovani su za povezivanje kao `f1@int@int*@int` i `f2@double@X*`, respektivno (ime f-je – povrtani tip – tipovi parametara).

Korišćenjem datih sistemskih poziva realizovati „patrljke“ (*stub*) za ove dve funkcije koje program treba da poziva da bi pristupio njihovim implementacijama u modulima. U slučaju greške pozvati funkciju `handleError` koja ima isti potpis i ponašanje kao bibliotečna funkcija `printf`, samo što ispis šalje na standardni izlaz za greške i potom završava pozivajući proces.

```
int f1 (int*, int);
double f2 (X*);
```

Rešenje: