Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Integralni, 1/2025.
Datum: 21. 9. 2025.

Kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje dva sata. Dozvoljeno je korišćenje literature.

Zadatak 1	_____/10	*Zadatak 3*	_____/10
Zadatak 2	_____/10	*Zadatak 4*	_____/10

Ukupno: _____/40

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)

Neki mikroprocesor za ugrađene (*embedded*) sisteme ne podržava virtuelni adresni prostor niti preslikavanje virtuelnih u fizičke adrese, pa operativni sistem za ovaj procesor ne podržava procese, već samo niti, kako je to i karakteristično za ovakve sisteme. Međutim, ovaj procesor ima u sebi *jedinicu za zaštitu memorije* (*memory protection unit*, MPU) koja omogućava zaštitu pojedinih regiona operativne memorije na sledeći način.

MPU omogućava da se u svakom trenutku u njemu definiše maksimalno 8 ulaza za isto toliko definisanih *regiona* operativne memorije koji se identifikuju brojevima 0..7. Za svaki region može se definisati početna adresa, veličina, kao i prava pristupa, redom navedenim parametrima operacije `MPU::setRegion` koja u odgovarajuće registre MPU upisuje odgovarajuće vrednosti kojima se region konfiguriše. Prava pristupa se definišu postavljanjem odgovarajućih bita za koje su definisane bit-maske odgovarajućim simboličkim konstantama: `O_KX`, `O_KRD`, `O_KWR`, `O_KRW` za dozvolu izvršavanja, čitanja, upisa, odnosno i čitanja i upisa, respektivno, u sistemskom (kernel) režimu rada procesora, kao i odgovarajuće konstante `O_UX`, `O_URD`, `O_UWR`, `O_URW` za korisnički režim rada procesora. Funkcija `MPU::invalidateRegion` invaliduje dati ulaz u MPU.

MPU proverava pristup memoriji tako što za datu generizanu adresu traži prvi definisani region (redom po brojevima ulaza) u koji data adresa ulazi (regioni se mogu i preklapati) i na osnovu prava pristupa definisanih za taj region odlučuje o dozvoli pristupa toj adresi.

Kernel operativnog sistema za ovaj procesor uvek organizuje šest regiona, numerisanih redom 0..5: kod, podaci i stek kernela, zatim kod, zajednički podaci svih niti i stek tekuće korisničke niti. Osim toga, kernel omogućava da svaka nit ima i svoj „privatan“ region za podatke, ukoliko ga je alocirao, koji je dozvoljen za pristup samo toj niti; ovaj region smešta u ulaz 6 MPU-a.

U klasi `Thread`, koja implementira nit u ovom kernelu, postoje sledeće nestatičke operacije:

- `void* getStackStart()`: vraća početnu adresu memorijskog prostora za stek date niti;
- `void* getStackSize()`: vraća veličinu memorijskog prostora za stek date niti;
- `void* getPrivDataStart()`: vraća početnu adresu memorijskog prostora za privatne podatke date niti, odnosno 0 ako ovaj prostor nije alociran;
- `void* getPrivDataSize()`: vraća veličinu memorijskog prostora za privatne podatke date niti.

Implementirati operaciju `switchMemContext` koja menja memorijski kontekst i koju kernel poziva kada menja kontekst niti i procesor predaje niti koja je data kao parametar.

```
MPU::setRegion (short regionNo, void* startAddr, size_t size, int prot);
MPU::invalidateRegion (short regionNo);
void switchMemContext (Thread* toRun);
```

Rešenje:

2. (10 poena)

Procesor poseduje dva pokazivača vrha steka: registar SP se koristi u neprivilegovanom režimu, a registar SSP u privilegovanom režimu rada procesora. Prilikom izvršavanja instrukcije sistemskog poziva, obrade izuzetka ili spoljašnjeg prekida, procesor najpre stavlja sadržaj registara PC i PSW, tim redom, na stek na čiji vrh ukazuje SSP, a potom prelazi u rutinu za obradu; SP se tom prilikom ne menja.

Kernel čuva kontekst tekućeg procesa na sistemskom delu steka koji je alociran za svaki proces, a za izvršavanje svog koda za obradu sistemskog poziva koristi poseban, jedinstven stek koji pripada samo kernelu. Na taj stek ukazuje globalna promenljiva čija je adresa predstavljena simboličkom konstantom `kernelSP`. U strukturi PCB postoji polje za čuvanje vrednosti vrha steka tekućeg procesa, čiji je pomeraj u odnosu na početak PCB-a definisan simboličkom konstantom `offsSP`. Pokazivač na tekući proces je u globalnoj promenljivoj na adresi `running`. Procedura kernela koja obrađuje sistemski poziv i nakon obrade poziva u `running` upisuje adresu PCB-a novog tekućeg procesa je `handle_sys_call`.

Na asembleru procesora picoRISC napisati proceduru za obradu sistemskog poziva i promenu konteksta, tako da za obradu sistemskog poziva koristi stek kernela.

Rešenje:

3. (10 poena)

Dole je dat interfejs drajvera blokovski orijentisanog uređaja („diska“). Drajver obavlja ulazno-izlazne operacije sa diskom pomoću DMA kontrolera. Operacijom `setIVTE` drajveru se zadaje dodeljen broj ulaza u IVT preko kog će DMA signalizirati završetak zadate operacije; drajver ovaj broj upisuje u odgovarajući registar DMA kontrolera. Operacija sa DMA pokreće se pozivom operacije `start` sa zadatim brojem bloka na disku `blkNo`, za prenos bloka podataka na zadatoj adresi `buffer` i za odgovarajući smer (`rdwr`, 0 za čitanje, 1 za upis). Nakon završenog prenosa, DMA kontroler generiše prekid sa zadatim brojem ulaza u IVT, a nakon toga status završene operacije može se očitati pozivom operacije `getStatus`.

Mehanizam prekida omogućava da se u IVT, pored pokazivača na prekidnu rutinu, zada i parametar tipa `void*` koji odgovara svakom ulazu. Taj parametar se dostavlja prekidnoj rutini prilikom obrade prekida u tom ulazu. Ulaz u IVT postavlja se na zadatu rutinu `isr` operacijom `Interrupts::initIVT`.

Implementirati klasu `BlockDevice`, zajedno sa odgovarajućom prekidnom rutinom, koja uporednim korisničkim procesima pruža usluge prenosa sa blokovskim uređajem. Pri inicijalizaciji, objektu ove klase dostavlja se broj ulaza u IVT koji je dodeljen datom uređaju, kao i drajver tog uređaja. Tada treba inicijalizovati drajver tim brojem ulaza, ali i postaviti vektor u IVT. Procesi zadaju operacije sa diskom pozivom operacija `read` i `write`. Ove operacije treba da se izvršavaju jedna po jedna, kako ih procesi zadaju, i da vraćaju status završene operacije. Sinhronizaciju obavljati semaforima koje implementira klasa `Semaphore` sa interfejsom kao u školskom jezgru.

```
class DeviceDriver {
public:
  virtual void setIVTE(IVTNo ivte);
  virtual void start (BlkNo blkNo, void* buffer, int rdwr);
  virtual int getStatus ();
};

void Interrupts::initIVT (IVTNo ivte, void(*isr)(void*), void*);

class BlockDevice {
public:
  BlockDevice (IVTNo ivte, BlkDeviceDriver* drv);

  int read  (BlkNo blkNo, void* buffer);
  int write (BlkNo blkNo, void* buffer);
};
```

Rešenje:

4. (10 poena)

Dat je podsetnik na neke osnovne Unix komande:

- `cat`: iz svakog fajla koji je naveden kao argument ove komande, redom kojim su oni navedeni, učitava znakove i ispisuje ih na standardni izlaz; ukoliko nema argumenata, znakove učitava sa standardnog ulaza (dok ne naiđe na znak EOF koji se na konzoli signalizira pritiskom tastera Ctrl+D).
- `ln` *`src_file dst_file`*: pravi novu tvrdu vezu (ulaz u direktorijumu) na postojeći fajl *src_file* u odredištu definisanom sa *dst_file*; opcija -s radi isto to, samo što pravi meku (simboličku) vezu.
- `rm`: briše fajl ili direktorijum ili vezu zadat kao argument.

Izvršavaju se sledeće komande po datom redosledu; napisati te komande ili odgovoriti na postavljeno pitanje:

- U postojećem poddirektorijumu `b` tekućeg direktorijuma napraviti nov tekstualni fajl `bdoc` čiji će sadržaj biti isti kao sadržaj postojećeg tekstualnog fajla `adoc` u roditeljskom direktorijumu tekućeg:

_____

- U roditeljskom direktorijumu tekućeg direktorijuma napraviti tvrdu vezu pod nazivom `hdoc` na fajl napravljen u koraku 1:

_____

- U postojećem poddirektorijumu `a` tekućeg direktorijuma napraviti meku vezu pod nazivom `sdoc` na vezu napravljenu u koraku 2:

_____

- Obrisati gorepomenute `hdoc` i `adoc`:

_____

- Šta će nakon ovoga ispisati komanda `cat a/sdoc`?

_____

- Nakon izdate komande `rm b/bdoc`, šta će ispisati komanda `cat a/sdoc`?

_____