1. (2) Slojevita izgradnja operacijskih sustava (npr. *arch/kernel/api/programs*) ima svoje prednosti i nedostatke. Navedite neke prednosti i nedostatke.
2. (2) Ako je zastavica `IF` (*Interrupt Flag*) obrisana, hoće li instrukcija `INT 33` generirati prekid (koji će se prihvatiti i obraditi)? Zašto?
3. (2) Opišite „upravljanje“ UI jedinicom koje omogućuje jezgra operacijskog sustava. Koji su sve elementi upravljanja potrebni?
4. (2) Što su to *callback* funkcije? Gdje se one koriste u jezgri operacijskog sustava?
5. Globalne varijable pojednostavljuju komunikaciju između različitih funkcija (nije potrebno prenositi te podatke kao parametre funkcija). Međutim, ponekad one stvaraju probleme.

   a) (2) Neki od njih se manifestiraju kad je kod u više datoteka. Koji su to problemi i kako se rješavaju?

   b) (2) Globalne varijable su osobit problem u višedretvenom okruženju. Koji se problemi ovdje javljaju i kako se rješavaju?
6. (2) Koja je osnovna razlika u komunikaciji preko reda poruka naspram cjevovoda? Skicirati sučelja za oba principa komunikacije.
7. (2) Kako ostvariti zaštitu jezgre od procesa? Opisati mehanizme, mogućnosti te zahtjeve prema sklopovlju.

8. Izvorni kod programske komponente nekog sustava smješten je u direktorije `boot`, `kernel` i `programs`.
   a) (1) Napisati skriptu povezivača koja će pripremiti sustav za početnu adresu 0x10000.
   b) (1) Ponoviti a) ali tako da na početku bude podatkovni dio (konstante i inicijalizirane globalne varijable) iz izvornih kodova iz `boot` direktorija.
   c) (1) Ponoviti a) ali tako da se instrukcije i konstante smjeste na adresu 0x10000 (ROM) a sve ostalo da bude pripremljeno za adresu 0x20000 (RAM), iako će biti početno učitano u ROM iza instrukcija i konstanti.
   d) (2) Ponoviti c) ali tako da su instrukcije i podaci iz `boot` i `kernel` direktorija pripremljeni za apsolutne adrese (navedene u c)) a programi iz `programs` pripremljeni u relativnim adresama (za adresu 0). U skriptu dodati potrebne oznake koje iskoristiti u funkciji `boot()` koja kopira podatke iz ROM-a u RAM, i po kopiranju poziva `startup()`.
9. (3) Neko ugrađeno računalo ima brojilo koje odbrojava od zadane vrijednosti do nule s frekvencijom od 1 MHz. Najveća vrijednost koja stane u brojilo jest `CNT_MAX`. Kada brojilo dođe do nule izaziva prekid broj `CNT_IRQN`, u brojilo automatski učitava `CNT_MAX` te nastavlja s odbrojavanjem. Prekidni podsustav nudi sučelje `void register_interrupt ( irq_num, handler )`. Neka se sadržaj brojila može dohvatiti sa `int cnt_get()` a postaviti sa `void cnt_set (int value)`. Ostvariti sustav praćenja vremena, tj. sučelje `void dohvati_trenutno_vrijeme ( int *sec, int *usec )` koje vraća trenutno vrijeme s preciznošću od mikrosekunde (uz pretpostavku brzog procesora, tj. zanemarenja trajanja izvođenja funkcija za dohvat vremena). Definirati sve potrebne strukture podataka i funkcije.
10. (4) Ostvariti jednostavno raspoređivanje dretvi sučeljem `void schedule ()` koje dretve izravno pozivaju kad žele prepustiti procesor drugim dretvama (ako takvih ima). Pretpostaviti da nema mogućnosti blokiranja dretvi, tj. sve se dretve (osim aktivne) nalaze u redu pripravnih složene prema redu prispijeća (FIFO). Definirati potrebnu strukturu te napisati kod navedene funkcije (i sve ostale pomoćne funkcije). Koristiti C i po potrebi strojne instrukcije (Intel ili ARM arhitekture ili slične onima koje postoje u tim arhitekturama).

11. (4) Ostvariti programsku komponentu za semafor koji prikazuje vrijeme, rezultat te dodatne informacije na zahtjev korisnika. Neka je „uobičajeni“ režim rada: 10 sekundi prikaz trenutnog vremena (minute:sekunde), 5 sekundi prikaz rezultata (domaći:gosti). Na zahtjev (prekid `IRQ_MSG`) treba prikazati traženu poruku (koja se dohvaća s `char *dohvati_poruku ()`) na 15 sekundi te po isteku tih 15 sekundi nastaviti s uobičajenim režimom.
    Na raspolaganju stoje sučelja (NISU jezgrine funkcije!):
    `void registriraj_prekid ( int irqn, void (*funkcija)() )` - za registraciju prekida,
    `void zabrani_prekidanje ()` - za zabranu prihvata prekida naprava,
    `void dozvoli_prekidanje ()` - za dozvolu prihvata prekida naprava,
    `int ispisi ( char *format, ... )` - za ispis („standardna“ *printf* funkcija),
    `void dohvati_poruku ( char *poruka )` - za dohvat poruke koju treba ispisati (na zahtjev),
    `void dohvati_vrijeme ( int *min, int *sek )` - za dohvat trenutnog vremena (koje se nezavisno ažurira – nije potrebno ostvariti, vrijeme kreće od 0:0 na početku igre),
    `void dohvati_rezultat ( int *domaci, int *gosti )` - za dohvat trenutnog rezultata, te
    Korištenjem navedenih sučelja (nema drugih!) ostvariti upravljanje semaforom u funkciji `void semafor ()`. Navesti sve potrebne strukture podataka i pomoćne funkcije. Preciznost vremena i rezultata treba biti unutar sekunde.

12. (4) Ostvariti sustav detekcije potpunog zastoja nad mehanizmom semafora. Pretpostaviti da su sve dretve u listi `lista_t dretve` te da u opisniku dretve `(dretva_t)` postoje elementi: `int stanje` (AKTIVNO, PRIPRAVNO, BLOKIRANO), `lista_t *red` - kazaljka na red u kojem se dretva nalazi. U opisniku semafora `(sem_t)` postoje elementi: `int vrijednost` (trenutna vrijednost semafora) te `lista_t red` – red za blokirane dretve. Svi opisnici semafor nalaze se u listi `lista_t semafori`. Za rad s redom postoji sučelje: `void *red_prvi ( red_t *red )` i `void *red_iduci( void *element )`. Kad je red prazan, odnosno, kad nema idućeg elementa funkcije vraćaju NULL. U funkciji `Čekaj_Semafor ( sem_t *sem )`, nakon blokiranja dretve poziva se funkcija: `provjera_potpunog_zastoja ()` u koju treba dodati kod detekcije potpunog zastoja. Potpuni zastoj definiramo kao stanje kada se SVE dretve nalaze u redu nekog semafora (dretve mogu biti blokirane i zbog drugih razloga, ali tada nije nastupio potpuni zastoj). Pri detekciji potpunog zastoja pozvati `LOG ( ERROR, 'Potpuni zastoj' )`.

13. (4) Prikazati (skicirati) postupak ostvarenja jezgrine funkcije `int povecaj ( int *broj )` koja povećava vrijednost na adresi `broj` za jedan u sustavu u kojem se programi izvode u logičkom adresnom prostoru (adresni prostor procesa kreće od adrese 0). Prikazati i funkciju i pomoćne funkcije i strukture podataka. Odabir načina prijenosa parametara u jezgrinu funkciju je proizvoljan (npr. stog ili zasebne varijable u dretvi).