1. (2) Ostvariti prekidni podsustav u sustavu kod kojeg su svi pristupni sklopovi naprava spojeni na isti prekidni signal koji se dovodi do računala. Za svaki pristupni sklop naprave poznata je adresa upravljačkog registra. Čitanjem vrijednosti registra dobiva se stanje pristupnog sklopa: vrijednost 1 označava da je taj pristupni sklop generirao zahtjev za prekid koji črks ns obradu (0 inače). Pisanjem vrijednosti 0 u taj registar javlja se sklopu da je njegov prekid prihvaćen, te će sklop ugasiti svoj zahtjev za prekid (prekidni signal) i nastaviti s radom (i opet generirati zahtjeve za prekid po potrebi). Pisanjem vrijednosti 1 u taj sklop nalaže se sklopu da ne izaziva zahtjeve za prekid (iduća 0 će to opet omogućiti). Prekidni podsustav treba imati sučelja:
   `int register ( void *control_register, void (*interrupt_handler)() );` te
   `int unregister ( void *control_register );`.
   Pored navedenih funkcija prikazati i funkciju `void int_handler ();` koja se poziva pri svakom prekidu (to je već podešeno pri inicijalizaciji sustava), kao i sve dodatne potrebne strukture.

2. (2) Za ostvarenje nadzornog alarma na raspolaganju stoji 32-bitovno brojilo koje odbrojava (frekvencijom od 250 kHz) od zadane vrijednosti do 0, kada izaziva prekid RESET. Upisivanjem vrijednosti u brojilo, ono započinje s odbrojavanjem od te vrijednosti. Međutim, ako se upiše 0 brojilo se deaktivira (ne broji i ne izaziva prekid). Brojilo je dohvatljivo na adresi 0x1000. Napisati slijedeće funkcije za rad s nadzornim alarmom:
   ```
   int wdt_start ( int miliseconds );
   int wdt_stop ();
   int wdt_signal ();
   ```
   Pretpostaviti da pri inicijalizaciji zadajemo vrijeme u milisekundama i da ono neće premašiti vrijednost 1000000.

3. (3) Za neku napravu napisan je skup funkcija:
   ```
   int init ( int irq_num );
   int get_status ();
   int send ( void *buffer, size_t size );
   int recv ( void *buffer, size_t size );
   ```
   Parametar `irq_num` definira koji će prekid naprava izazvati pri primitku novih podataka ili po dovršenju slanja, parametri `buffer` i `size` definiraju međuspremnik za slanje i primanje podataka. Funkcije `send` i `recv` vraćaju veličinu poslanih/primljenih podataka. Funkcija `get_status` vraća 0 ako naprava nema posla, 1 ako je u tijeku slanje ili primanje podataka (napravu treba pustiti da obavi slanje/primanje prije nego li se koriste funkcije `send/recv`). Korištenje ove naprave operacijski sustav treba nuditi kroz sučelje:
   ```
   int devx_init ();
   int devx_send ( void *buffer, size_t size );
   int devx_recv ( void *buffer, size_t size );
   ```
   Pokazati ostvarenje tih funkcija, kao i funkcije koja se poziva na prekid naprave. Pretpostaviti postojanje prekidnog podsustava (funkcije `register_interrupt_handler`). Nadalje, za komunikaciju s napravom rezervirati i koristiti dodatne međuspremnike (ako naprava trenutno već šalje podatke, nove podatke za slanje staviti u međuspremnik koji će se proslijedit napravi kad bude gotova s prethodnim poslom; slično i po čitanju – podaci se prenose u međuspremnik, a iz njega uzimaju sa `recv`).

4. (2) Za ostvarenje dinamičkog upravljanja spremnikom (`malloc/free`) potrebne su neke funkcije i strukture podataka. Skicirajte te funkcije i strukture podataka.

5. (2) Napisati makro `MAX(A,B,C)` koji će veću vrijednost od `A` i `B` staviti u `C` (sam makro ne vraća vrijednost). Pretpostaviti da su parametri tipa `int`, ali i da mogu biti izrazi. Primjeri za koje makro treba raditi ispravno (ali i ne promijeniti semantiku programa!!!):
```
MAX ( 5, a*b, max );
MAX ( a, a+b+c, c );
MAX ( read(f1, b1, n1), read(f2, b2, n2), c );
```
Navedeni pozivi mogu biti nezavisni ili dio `if/else/*` grana, npr.
```
if ( x > 0 )
    MAX (a, b, c );
else
    MAX ( fun1(a), fun2(b), c );.
```

6. (2) Opišite postupak zamjene jedne dretve drugom bez korištenja mehanizma prekida (kako je to napravljeno u `Chapter_07_Threads`).
Skicirajte u pseudokodu ili asembleru funkciju `zamijeni_dretvu ( stara, nova )`.

7. (1) Navedite barem šest elemenata opisnika dretve prema `Chapter_07_Threads` ali opisno (nisu potrebna imena koja se tamo koriste) i za svaki njegovu svrhu.

8. (1) Proširenje sučelja za monitore funkcijom:
`int pthread_mutex_timedlock ( pthread_mutex_t *m, timespec_t *t );`
(dio 3. laboratorijske vježbe) zahtijeva, osim nekih novih funkcija, promjene i u postojećim funkcijama i strukturama podataka. Opišite te promjene.

9. (1) Mehanizam signala omogućava asinkronu "komunikaciju" među dretvama (i OS-a), ali predstavlja i mnoge probleme za ostvarenje jezgre OS-a. Navedite i opišite neke od njih.

10. (2) Kako ostvariti zaštitu (jedne dretve od druge, jezgru od dretvi) u višedretvenom okruženju? Opišite potrebne mehanizme na razini sklopovlja te kako se oni iskorištavaju programski (u jezgri i programima).

11. (1) Pri upravljanju procesima koji koriste logičke adrese javljaju se neki novi problemi (osim korištenja sklopovlja za pretvorbu adresa pri izvođenju procesa). Koji su to problemi i kako se rješavaju?

12. (2) Neki ugrađeni sustav sastoji se od mikro jezgre čiji je kod u direktoriju `jezgra` te tri programa, svaki u svom direktoriju `prog1`, `prog2`, i `prog3`. Ukoliko se programi (procesi) izvode u logičkom adresnom prostoru (koristi se straničenje), napisati zajedničku skriptu za povezivača (linkera) koji će pripremiti jezgru za adresu 0x10000 (u jezgri se koriste fizičke adrese) te programe (koji se učitavaju odmah iza jezgre) za logičku adresu 0 (svaki se zasebno priprema za tu adresu, od svakog programa nastaje zasebni proces). Prilikom pokretanja programa (stvaranja novog procesa), jezgra mora kopirati dio učitanih programa (.data i .bss dijelove) na novu lokaciju. Stoga skripta za povezivača mora uključivati i dodatne varijable.