**OSUR, usmeni ispit**

1. (4) Navesti nekoliko bitnih razlika u pripremi (kôdiranju) i prevođenju „obična“ programa (za neki operacijski sustav) te pripremu i prevođenje programske potpore za ugrađeno računalo koja uključuje i OS i programe.

   Pri pripremi obična programa moguće se osloniti na razne biblioteke za komunikaciju s OS-om, napravama i ostalim programima (procesima/dretvama).

   Pri pripremi programske potpore za ugrađena računala treba napraviti i OS. Stoga je potrebna razina znanja znatno veća – treba poznavati sklopovlje, način rada OS-a (jezgre), priprema koda za specifičnu arhitekturu (ROM, RAM, …), ograničenja sklopovlja (odabrati pravi algoritam, snaći se s manje memorije, …)

2. (4) Kraće funkcije mogu se napraviti na nekoliko načina: korištenjem makroa (umjesto funkcije), označavanjem sa inline ili kao „normalne“ funkcije. Navesti kada jedno, drugo ili treće ima smisla (donosi neku korist / potrebno je napraviti baš tako).

   Makroi: ugrađuju je na mjesto poziva => **brži rad** (ne pozivaju se funkcije, prijenos parametara, …), ali kod postaje **veći** (treba više memorije za program)

   Inline funkcije: **slično kao i makroi** ugrađuju se na mjesto poziva. Međutim, postoji mogućnost da ih prevoditelj ipak ne ugradi (ovisno o naredbama koje mu damo), pa da budu kao obične funkcije.

   Obične funkcije: pozivaju se „klasično“ (argumenti na stog + CALL) => zbog toga **nisu toliko učinkovite**, ali se kod **ne multiplicira** kako kod makroa i inline funkcija.

3. (3) Ako je zastavica IF (engl. *interrupt enable flag*) obrisana, hoće li instrukcija INT 33 izazvati prekid (koji će se prihvatiti i obraditi)? Zašto?

   Instrukcija INT 33 generira zahtjev za prekid unutar procesora. Zastavica IF u registru stanja sprečava prihvat samo prekida koji dolaze izvana. Stoga se ovaj prekid **prihvaća**.

4. (3) Koja sve svojstva treba razmatrati pri izboru algoritma za dinamičko upravljanje spremnikom?

   **Složenost algoritma** (npr. O(1), O(n), O(log n)) utječe na predvidljivost ponašanja (da u nekim situacijama nam ne treba više vremena nego što očekujemo – ne stigne se nešto obaviti u očekivanom vremenu).

   **Fragmentacija** slobodnih blokova može onemogućiti posluživanje nekih zahtjeva.

5. (3) Što je to *povratna vrijednost funkcije*, a što *oznaka greške* (engl. error number)?

   Primjer:

```
a = funkcija(argumenti);
if (a == -1) { //a je povratna vrijednost, -1 najčešće označava grešku
   //analiza greške preko oznake greške, varijable/makroa errno
   if (errno == EINVAL) {
      //problem u argumentima
   }
}
```

6. (4) Signal se dretvi može poslati u „neočekivanom“ trenutku. Koji problemi mogu nastati zbog toga (kako/kada prihvatiti signal)?

   Očekivani trenutak je da je dretva nešto radila, bila aktivna ili pripravna.

   Neočekivani trenutak je kad je dretva blokirana, u nekom redu, obično kao posljedica poziva neke jezgrine funkcije (npr. read, sleep, sem_wait i sl.). U tom slučaju, na signal dretva može biti „probuđena“, obraditi signal, ali nakon toga se obično ne vraća u blokirano stanje nego funkcija vraća -1 i errno = EINTR. Druga mogućnost je da se signal zadrži dok se dretva ne odblokira (na uobičajeni način za to stanje dretve).

7. (4) Što su to *procesi*? Koje mehanizme dodatno pružaju u odnosu na "samo višedretvenost"?

   Procesi su mehanizam izolacije izvođenja dretvi (ili skupa dretvi) od jezgre OS-a i ostalih procesa. Ostvaruju se **izolacijom** adresnog prostora (npr. straničenjem) te izvođenjem u **korisničkom** načinu rada. Na taj način eventualna greška jedne dretve može biti lokalizirana, utječe samo na taj proces a ne i ostatak sustava.