Pismeni ispit iz predmeta *Operacijski sustavi za ugrađena računala*, 20. 6. 2011.

1. (8) Sustav za koji se programira sastoji se od ROM-a na adresi 0 te RAM-a na adresi 0x100000. Jedan projekt sastoji se od datoteka: `startup.c`, `mkernel.c`, `p1.c`, `p2.c` i `p3.c`. Pri prevođenju sve će se izlazne (.o) datoteke sastojati samo od `.text`, `.rodata`, `.data` i `.bss` odjeljaka. Izlazna datoteka `kernel.bin` koja nastaje povezivanjem svih izlaznih objektnih datoteka posebnim se alatom upisuje u ROM.

   Napisati program u `startup.c` koji treba prekopirati podatke jezgre i sve iz programa (`.rodata`, `.data` i `.bss` iz `mkernel.o` te `.text`, `.rodata`, `.data` i `.bss` iz `p*.c`) u radni spremnik (RAM) na adresu 0x100000. Pritom, obzirom da se koristi straničenje *programe* (svaki zasebno) treba pripremiti za adresu 0 (a podatke jezgre za adresu 0x10000, kôd za ROM gdje se prvotno i učitava).

   Napisati skriptu `skripta.ld` za povezivača (*linkera*) tako da navedeno bude moguće.

2. (6) Napisati `Makefile`, `zaglavlje.h` te nadopuniti datoteke `int.c` i `double.c` tako da sustav radi ispravno.

`int.c`:
```
int suma = 0;
int brojac = 0;
inline int zbroji ( int a, int b )
{ return a + b; }
inline int podijeli ( int a, int b )
{ return a / b; }
inline void dodaj_i ( int broj )
{ suma = zbroji ( suma, broj ); brojac++; }
inline int sv_i ()
{ return podijeli ( suma, brojac ); }
```

`prekid.c`:
```
#include „zaglavlje.h"
static int dohvati_pod_i ( int broj ) { … }
static double dohvati_pod_d ( int broj ) { … }

void prekidna_rutina ( int broj ) {
   if ( broj > 0 )
      dodaj_i ( dohvati_pod_i (broj) );
   else
      dodaj_d ( dohvati_pod_d (broj) );
}
```

`double.c`:
```
double suma = 0;
int brojac = 0;
inline int zbroji ( double a, double b )
{ return a + b; }
inline int podijeli ( double a, double b )
{ return a / b; }
inline void dodaj_d ( double broj )
{ suma = zbroji ( suma, broj ); brojac++; }
inline int sv_d ()
{ return podijeli ( suma, brojac ); }
```

`main.c`:
```
#include „zaglavlje.h"
int main () {
   dodaj_i ( 0 );
   dodaj_d ( 0 );
   inicijaliziraj_prekide ( prekidna_rutina );
   for ( ; ; ) {
      sleep (1);
      printf ( „%d:%lf\n", sv_i(), sv_d() );
   }
   return 0;
}
```

3. (8) Ostvariti podsustav za upravljanje vremenom, tj. samo osnovnu (i jedinu) jezgrinu funkciju `void k_postavi_alarm ( kada, akcija )`. Željena funkcionalnost ne uključuje naknadnu promjenu već postojećih alarma ili njihovo brisanje već se oni nakon aktivacije (nakon poziva zadane funkcije `akcija`) tada miču (tu funkcionalnost također ostvariti). Pretpostaviti da na raspolaganju stoje operacije za rad s listom (`list_init`, `list_dodaj_sort (list, elem, funkc_usporedbe)`, `list_prvi`, …) te sučelje *arch* sloja `arch_postavi_alarm ( kada, akcija )` (sučelje *arch* sloja pamti samo zadnji alarm, prethodno postavljeni se prebriše novim).

4. (3) Napraviti makro koji povećava dvije varijable za jedan, tj. popraviti idući makro tako da on bude uporabljiv u svim smislenim dijelovima programa: `#define INC(A,B) A++;B++`

5. (3) Zadan je makro i njegov poziv iz kôda:
   definicija: `#define LOG(LEVEL, X) printf (#LEVEL „-%d", status->m_ ## X)`
   poziv iz kôda: `LOG ( ERROR, sent );`

   Napisati kako će izgledati kôd nakon preprocesorske obrade, tj. kako će se poziv zamijeniti s makroem (a prije prevođenja „čistog" C-a).

6. (4) U predlošcima za razne elemente korišteno je posebno oblikovano „sučelje", primjerice: `device_t`, `arch_timer_t`, i `arch_ic_t`. Kako se ta sučelja koriste u jezgri, a kako im se pristupa iz programa (prije *Chapter_8*)? Opišite osnovni princip (može na primjeru).

7. (8) Proširiti monitorske funkcije (i ostale potrebne podatkovne strukture monitora i/ili opisnika dretve) da se omogući mehanizam rekurzivnog zaključavanja. U nastavku su navedene monitorske funkcije (iz skripte).

```
j_funkcija Zaključaj_monitor ( m )
{
      ako ( Mon[m].v == 1 )
      {
               Mon[m].v = 0;
      }
      inače {
               stavi_u_red ( Mon[m].r, Aktivna_dretva );
               odaberi_aktivnu_dretvu();
      }
}
j_funkcija Otključaj_monitor ( m )
{
      ako ( red Mon[m].r je prazan )
      {
               Mon[m].v = 1;
      }
      inače {
               stavi_u_red ( Pripravne_dretve, uzmi_prvu_iz_reda ( Mon[m].r ) );
               odaberi_aktivnu_dretvu();
      }
}
j_funkcija Čekaj_u_redu_uvjeta ( m, red )
{
      pohrani_pripadajući_monitor ( Aktivna_dretva, m );
      stavi_u_red ( Red_uvjeta[red].r, Aktivna_dretva );

      ako ( red Mon[m].r je prazan )
      {
               Mon[m].v = 1;
      }
      inače {
               stavi_u_red ( Pripravne_dretve, uzmi_prvu_iz_reda ( Mon[m].r ) );
               odaberi_aktivnu_dretvu();
      }
}
j_funkcija Propusti_iz_reda ( red )
{
      ako ( red Red_uvjeta[red].r nije prazan )
      {
               prva = uzmi_prvu_iz_reda ( Red_uvjeta[red].r );
               m = dohvati_pripadajući_monitor (prva);

               ako ( Mon[m].v == 1 ) //dretva se propušta u monitor
               {
                        Mon[m].v = 0;
                        stavi_u_red ( Pripravne_dretve, prva );
                        odaberi_aktivnu_dretvu();
               }
               inače { //neka druga dretva je u monitoru, treba pričekati da izađe
                        stavi_u_red ( Mon[m].r, prva );
               }
      }
}
```

8. (3) Navesti prednosti i nedostatke izravnog pozivanja jezgrinih funkcija (kao što je to do *Chapter_7*) naspram poziva preko programskih prekida (*Chapter_8*).

9. (3) Navesti probleme dinamičkog upravljanja spremnikom (dinamičko upravljanje kao što je ostvareno u `malloc/free`, ne upravljanje procesima). Navesti nekoliko algoritama te njihova dobra i loša svojstva.

10. Navesti potrebnu jezgrinu strukturu podataka za opis procesa ako se za upravljanje spremnikom koristi:

    a) (2) segmentacija,

    b) (2) straničenje.