Operacijski sustavi za ugrađena računala – međuispit, 23. 4. 2015.

1. U nekoj arhitekturi gdje procesorska riječ ima 8 bita treba napraviti tip podataka sa 16 bita (za koji je najveći broj `0xFFFF`). Za njegovo ostvarenje korištene su dvije varijable: `unsigned char a1, a2`. Prva `a1` sadrži nižih 8 bita a druga `a2` viših. Napisati makroe `INC()`, `DEC()` koji povećavaju/smanjuju 16-bitovni podatak (pohranjen u `a1` i `a2`) za jedan. Makroi ne trebaju vraćati vrijednost. Pretpostaviti prelaske granice s 255 u 0 pri zbrajanju te s 0 na 255 pri oduzimanju.

```
#define INC() do { a1++; if (a1 == 0) a2++;} while(0)
#define DEC() do { a1--; if (a1 == 0xffff) a2--;} while(0)
```

2. Zadane su dvije slične strukture `struct first` i `struct second` koje imaju jedan element za istu namjenu, ali se različito zove: `size_t first_size` za prvu i `size_t second_size` za drugu. Napisati makro `SET_SIZE(obj,struct_name,size)` koji će postaviti veličinu objekta u odgovarajući element strukture. Npr. za varijablu `A` tipa `struct first` poziv `SET_SIZE(A,first,25)` se treba prevesti u `A.first_size = 25`. Povratna vrijednost makroa je postavljena veličina.

```
#define SET_SIZE(obj,struct_name,size) ((obj).struct_name ## _size = 25)
```

3. Napisati makro `NONZERO(a,b,c,d)` koji će za četiri ulazna parametra vratiti prvu vrijednost različitu od nule (vrijednost `a` ako `a!= 0`, `b` ako je `a == 0` i `b!=0`, itd). Ako takva vrijednost ne postoji, vratiti -1.

```
#define NONZERO(a,b,c,d) (a?a:(b?b:(c?c:(d?d:-1))))
```

4. Neki sustav zadan s mnoštvo datoteka s izvornim kodom treba pripremiti za ugrađeni sustav. Sliku sustava treba učitati na adresu `ROM=0x100`, ali se po pokretanju tog sustava sve treba prebaciti na `RAM=0x100000`. Prebacivanje treba obaviti funkcijom `init()` u datoteci `init.c` (koja se jedina priprema za izvođenje iz `ROM`a, nema nikakvih dodatnih pomoćnih funkcija na raspolaganju). Napisati potrebnu skriptu povezivača (`ldscript.ld`) te sadržaj datoteke `init.c`. Pretpostaviti da će pri prevođenju nastati samo odjeljci `.text, .rodata, .data i .bss`.

```
ldscript.ld:
ROM=0x100;
RAM=0x100000;
SECTIONS {
   .init ROM: { init.o (*) }
   start = ROM + SIZEOF(.init);
   .ostalo RAM: AT ( start)
   { * (.text .rodata .data .bss) }
   size = SIZEOF(RAM);
}

init.c:
void init() {
   extern char start, size;
   char *rom = &start, *ram = 0x100000;
   int i;
   for ( i = 0; i < (size_t) &size; i++ )
         ram[i] = rom[i];
}
```

5. U nekom sustavu kada neka naprava izazove prekid ona na adresu `0x4000+irq` stavlja jedinicu, gdje `irq` određuje i prioritet naprave (manji broj predstavlja veći prioritet). Pri prihvatu prekida programski treba proći taj dio spremnika i ustanoviti koje naprave koja traže prekid (i obrisati te jedinice). Definirati potrebnu strukturu podataka te ostvariti sučelje za registraciju funkcije za obradu pojedinog prekida `void register(int irq, void *handler)` i funkciju `void interrupt_handler()` koja će se pozvati po detekciji zahtjeva za prekid. Obrada prekida treba biti prema prioritetima – programski ostvariti prihvat i obradu prekida prema prioritetima (sama obrada treba se moći prekidati). Pretpostaviti da funkcijama `enable_interrupts()` i `disable_interrupts()` se može omogućiti i zabraniti prekidanje te da u sustavu ima 30 naprava.

```
#define N 30
void (*hnd[N])(int irq);
int rq[N], scp[N], cp = N+1;
char *base = 0x4000;

void register ( int irq, void *handler ) {
  hnd[irq] = handler;
}
void interrupt_handler() {
  int i, hprq = cp;
  for ( i = 0; i < N; i++ ) {
        if ( base[i] == 1 ) {
              rq[i] = 1;
              base[i] = 0;
              if ( i < hprq )
                    hprq = i;
        }
  }
  while ( hprq < cp ) {
        rq[hprq] = 0;
        scp[hprq] = cp;
        cp = hprq;
        enable_interrupts();
        hnd[hprq](hprq);
        disable_interrupts();
        cp = scp[hprq];
        for ( hprq = 0; hprq < cp; hprq++ )
              if ( rq[hprq] )
                    break;
  }
}
```

6. Neki ugrađeni sustav posjeduje 64-bitovno brojilo na adresi `0x500` koje odbrojava od početne vrijednosti `0xffffffffffffffff` prema nuli s frekvencijom od s 1 MHz i koje se može samo čitati. Ostvariti funkcije `long get_time()`, `set_time(long new_time)`, `void delay(long period)` i `void delay_until(long time)` koje dohvaćaju sat, postavljaju sat, odgađaju program za zadani broj mikrosekundi te odgađaju program dok ne dođe zadano vrijeme. Vrijeme (sat) neka bude izraženo u mikrosekundama. Pri pokretanju početna vrijednost sata treba biti 0. Odgodu ostvariti radnim čekanjem.

```
#define CNT    *((long*)0x500)
long zero =    0xffffffffffffffffL;
long get_time () {
  return zero - CNT;
}
void set_time ( long new_time ) {
  zero = new_time + CNT; //ignoring overflow
}
void delay ( long period ) {
  long now = CNT;
  while ( now - period < CNT )
        ;
}
void delay_until ( long time ) {
  while ( time > zero - CNT )
        ;
}
```

init

7. Globalna varijabla `cnt` mijenja se u funkciji koja se poziva iz obrade prekida, a koristi i u ostalim funkcijama. Koju je ključnu riječ poželjno staviti ispred deklaracije (`int cnt;`)?

   `volatile`

8. Navesti prednosti korištenja `Makefile`-a (u usporedbi s prevođenjem kod kojeg se sve zastavice i datoteke navode u komandnoj liniji).

   brže zadati naredbu
   brže prevođenje: ne prevode se datoteke koje nisu mijenjane od zadnjeg prevođenja
   različite postavke za prevođenje različitih datoteka

9. Koja sve svojstva algoritama za dinamičko upravljanje spremnikom treba razmatrati u okruženju ugrađenih sustava i sustava za rad u stvarnom vremenu?

   složenost dodjele/oslobađanje
   fragmentacija

10. Navesti prednosti i nedostatke slojevite podjele operacijska sustava (njegova izvorna koda) na sloj apstrakcije sklopovlja (HAL, `arch`), sloj jezgre (`kernel`), sloj sustavskih funkcija (`api`) te aplikacijski sloj (`programs`).

    + podjela, jasnoća, lokalnost, zaštita
    - učinkovitost (poziv mora proći kroz više slojeva)