7. Neki pisač je spojen na računalo preko pristupna sklopa koji se u računalu upravlja sa dva registra: podatkovnim na adresi POD te upravljačkim na adresi UPR. Spremnost za prihvat podataka pisač iskazuje postavljanjem nule u UPR. Bilo koja druga vrijednost označava da pisač trenutno ne može prihvaćati nove podatke za ispis. Napisati upravljački program za pisač koji će u slučaju da pisač ne može prihvaćati nove znakove koristiti i međuspremnik veličine 64 KB. Sučelja koja upravljački program treba imati su:

```
void *inicijaliziraj ();  //stvoriti potrebnu strukturu,
//ono što će biti slano kao zadnji parameter u idućim funkcijama
int pošalji ( void *podaci, size_t duljina, void *naprava );
int pročitaj ( void *podaci, size_t duljina, void *naprava );
int dohvati_status ( void *naprava );
```

Pretpostaviti da će se poziv pošalji pozivati češće (s prva dva parametra poslavljena u NULL, 0) tako da bi se podaci koji su u međuspremniku mogli ponovno pokušati poslati na pisač.

8. U sustavu koji koristi virtualnu memoriju (logičke adrese) program treba posebno pripremiti. Instrukcije treba pripremiti za logičku adresu ADR_INST, konstante za adresu ADR_KONST te podatke za ADR_PODACI. Polja `char _za_stogove_[VEL_STOGA]` i `char _pomocno_[VEL_POMOCNO]` koji se kao jedini podaci nalaze u datoteci stog.c treba postaviti jedno za drugim, tako da završavaju na adresi KRAJ_PROCESA. Napisati skriptu povezivača kojom će se to moći napraviti. Po potrebi može se napisati i datoteka stog.c.

9. Ostvariti funkciju `zamijeni_dretve ( opisnik *a, opisnik *b )` koja treba pohraniti kontekst prve dretve, obnoviti kontekst druge i s njom nastaviti raditi. Neka procesor ima registre opće namjene R0–R15, programsko brojilo PC, statusni registar SR te kazaljku stoga SP. Opisnici dretvi imaju samo jedan element x. U asembleru pristupiti varijablama izravno (a->x, b->x kao i u C-u). Kontekst se može pohraniti na stog dretve ili u posebno mjesto (npr. na a->x). Popis potrebnih instrukcija s kratkim opisom:

```
PUSHA; POPA - spremanje/obnavljanje registara R0-R15 na/sa stog-a
PUSHF; POPF - spremanje/obnavljanje registra stanja na/sa stog-a
PUSH Rx; POP Rx - spremanje/obnavljanje registra Rx na/sa stog-a
LDR reg, adr - reg = [adr]
STR reg, adr - [adr] = reg
MOV reg, const - reg = const, const može biti i labela
CALL labela - poziv potprograma (povratna adresa na stogu)
RET - povratak iz potprograma (obnavljanjem adrese sa stoga)
JMP labela - skok na adresu
```

10. U neki sustav potrebno je dodati praćenje vremena zasebno po pojedinoj dretvi: koliko se vremena potrošilo na izvođenje instrukcija dretve (user), koliko se vremena potrošilo u jezgrinim funkcijama pozvanim od strane dretve (sys) te koliko je ukupno vremena proteklo od kada je dretva pokrenuta (real). Uz pretpostavku dovoljno preciznog sata (koji održava operacijski sustav) i funkcije za dohvat tog sata (npr. k_sys_time()) opisati kako bi se ostvarilo navedeno praćenje (što treba dodati, gdje, ...).

[redacted] ispit iz predmeta: *Operacijski sustavi za ugrađena računala* 4. 7. 2017.

(Svi zadaci se boduju s do 5 bodova.)

1. Zadani makro za zbrajanje dva kompleksna broja ima nekoliko nedostataka:

   `#define CADD(A, B, Z) { Z->x = A->x + B->x; Z->y = A->y + B->y; }`

   Popraviti makro tako da se ti nedostaci poprave. Makro ne vraća vrijednost, ali se treba moći pozvati od bilo kuda i sa složenim parametrima (ali koji ne mijenjaju sustav ni parametre, npr. neće se zvati sa `CADD(CADD(A,B,B),B,C)`, ali može sa `CADD(a,b+c,d)`).

2. U nekom repozitoriju (dohvaćenom s `git clone`) treba napraviti neku promjenu, dodati neku funkcionalnost kroz promijene datoteka u direktoriju `lib`. Međutim, ako se pokaže da navedena nije (još) dobra želi se i dalje moći koristiti prijašnju inačicu sustava. Pokazati kako to postići korištenjem git-a (navesti i potrebne naredbe).

3. Skicirajte ostvarenje osnovnog nadzornog alarma (watchdog timer) korištenjem sučelja za upravljanje vremenom: `postavi_alarm (id, vrijeme, funkcija)` (`id` je opisnik alarma koji vraća navedena funkcija; prvi puta se poziva s `NULL`; `vrijeme` je zadano u mikrosekundama, `funkcija` je funkcija koju treba pozvati po aktivaciji alarma). Prikazati korištenje sučelja na primjeru.

4. Neki sklop za prihvat prekida ima 32 ulaza. Na svaki ulaz spojena je najviše jedna naprava (ne dijele se ulazi). Za svaki ulaz postoji upravljački/statusni registar (`PP[i]`). Bitom 0 omogućava se proslijeđivanje prekida prema procesoru kad ga se postavi u 1. Bit 1 postavlja naprava spojena na taj ulaz kada zahtijeva prekid. Ostali bitovi se ne koriste. Pretpostaviti da svi ulazi imaju jednak prioritet te da je vjerojatnost preklapanja dva različita zahtjeva zanemariva (uključujući obradu). Ostvariti prekidni podsustav.

5. Neko brojilo može odbrojavati frekvencijom od 1 MHz ili 1 kHz, ovisno o upisanoj vrijednosti u upravljačkom registru `BR_UPR`: ako se upiše 1 odbrojava s 1 MHz, inače s 1 kHz. Trenutna vrijednost brojila se može očitati i postaviti preko registra `BR`. Ostvariti podsustav za upravljanje vremenom (`dohvati_sat()`/`postavi_sat(sat)`) koje ostvaruje sat s preciznošću od 1 ms, te jedan alarm (`alarm(odgoda, funkcija)`) s rezolucijom od 1 μs (napisati kod u C-u). Kada se postavlja alarm mora se koristiti veća frekvencija jer jedino u tom načinu rada će brojilo po dolasku do nule izazvati prekid. Kada alarm nije postavljen, brojilo treba odbrojavati s nižom frekvencijom. Pretpostaviti da su brojilo i procesorska riječ 64 bitovni podaci (unsigned long) - "sve" vrijednosti stanu u njih, i sat se iskazuje u mikrosekundama.

6. U nekom sustavu potrebno je napraviti funkciju za zbrajanje 256 bitovnih cijelih brojeva. Za to treba koristiti polje cijelih brojeva tipa `int` (neka prvi element polja nosi najnižih x bitova). Broj bitova koji zauzima običan tip `int` nije unaprijed poznat (može biti 16, 32 ili 64). Ostvariti funkciju:

```
void zbroji ( int *a, int *b, int *c )
```

koja zbraja brojeve zapisane u polja `a` i `b` te rezultat pohranjuje u `c`.