Završni ispit iz Arhitekture računala 1 **15. lipnja 2012.** **Grupa na predavanjima: _____**

Prezime i ime (tiskanim slovima): _____ JMBAG: _____

Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Potpis: _____.

Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Završni ispit traje 150 minuta.

1.a. FRISC (1 bod) Koji će biti sadržaj registra R0, nakon izvođenja naredbe LOAD na FRISC-u za programski odsječak s desne strane: _____

```
ORG 130
DH 34, 56
DW 8558
...
LOAD  R0, (130)
```

1.b. FRISC (2,5 boda) Prilikom dekodiranja i izvođenja FRISC-ove naredbe **SUB R1, %D10, R2**, broj 10 se nalazi u 20 nižih bitova registra _____. Naredba koristi dva adresiranja koja se zovu _____ i _____. Prilikom izvođenja naredbe, na jedan ulaz ALU dovodi se podatak iz sklopa _____**,** a na drugi se ulaz dovodi podatak iz _____ (*napisati iz kojih dijelova procesora se dovodi podatak*).

1.c. FRISC (2 boda) Prekidni sustav procesora FRISC sastoji se od priključaka: _____. Obzirom na mogućnost programske zabrane/dozvoljavanja prekida, procesor FRISC podržava _____ prekide na priključku/priključcima _____ , te _____ prekide na priključku/priključcima _____. U registru SR nalazi se prekidna zastavica: _____. Prilikom prihvaćanja prekida, procesor FRISC automatski sprema povratnu adresu (*gdje*): _____, a stanje registra SR sprema (*gdje*): _____.

1.d. FRISC (2 boda) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-GPIO zovu se: _____ i _____. Ovi priključci koriste se u načinima rada (*nabrojite kojim*): _____. FRISC-GPIO **ne može** postati spreman u načinu/načinima rada: _____.

~~Osim registra maske, na prvoj adresi sklopa GPIO nalaze se upravljački registri: _____. Kad šaljemo upravljačku riječ na tu adresu, GPIO zna u koji registar je želimo upisati na temelju (*čega?*): _____.~~ **(ovaj dio pitanja nije primjenjiv na novo gradivo)**. Ako želimo da GPIO (spojen na adresi FFFF4000) generira maskirajući prekid kad se na bilo kojem od 3 najniža bita postave nule, onda ga inicijaliziramo tako da pošaljemo podatak _____ na adresu _____.

2.a. ARM (**0,5 boda)** Procesor ARM izvodi odsječak programa s desne strane. Nakon izvođenja naredbe LDRB sadržaj registra R1 će biti: _____, a sadržaj registra R0: _____.

```
ORG 0
MOV R0, #1<8
LDRB R1, [R0,#4]!
ORG 100
DW 88776655, 44332211
```

2.b. ARM (0,5 boda) Procesor ARM izvodi naredbu LDMFD R13!, {R2,R1}. Ako je sadržaj registra R13 prije izvođenja naredbe bio 100_{10}, sadržaj registra R13 nakon naredbe će biti _____ , a registar R1 će se napuniti podatkom s memorijske lokacije _____.

2.c. ARM (1 bod) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: _____ , _____ i _____. Uvjetno izvođenje kod procesora ARM moguće je za (*koje?*) _____ naredbe.

2.d. ARM (**0,5 boda**) Za procesor ARM kod naredaba skokova moguća je pojava _____ hazarda. Negativni efekti ovog hazarda se u nekim procesorima umanjuju pomoću postupka _____ .

2.e. ARM (**1 bod**) Procesor ARM9 uvodi _____ arhitekturu memorijskog pristupa. Time se izbjegava _____ hazard koji postoji kod ARM7. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave _____ hazarda čiji se negativni efekti umanjuju pomoću postupka _____.

OKRENUTI STRANICU! ☺

2.f. ARM (**1 bod)** Nakon uključenja procesor ARM 7 treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:_____ (*napisati postupak rješavanja!*)

```
   ORG 0
   MOV R1, #5
   ADD R0,R1,#2
   BIC R0, R0, #%b100
A  SUBS  R0,R0,#1
   BNE A
   EOR R0,R0,R0
```

2.g. ARM (*2 boda*) Kod ARM-a imamo prekide _____(A) i _____(B). Adresa prekidnog potprograma (A) je _____, a za prekid (B) adresa potprograma je _____. Za prekid (A) povratna adresa se sprema (*gdje*): _____, a za prekid (B) povratna adresa se sprema (*gdje*): _____. Povratak iz prekidnog potprograma (A) ostvaruje se naredbom _____, a iz (B) pomoću naredbe _____.

2.h. ARM (1 bod) Neka priručna memorija ima 16_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
10	
11	
25	
3F	

3. (*7 bodova*) U računalnom sustavu nalaze se procesor FRISC, DMA, CT (na ulaz CNT spojen je signal frekvencije 1 kHz) te dva sklopa PIO. Adrese vanjskih jedinica odaberite sami.

Procesor u glavnom programu treba neprestano čitati 8-bitne NBC podatke sa sklopa PIO1 (koji radi u uvjetnom načinu rada). Svaki podatak treba pomnožiti sa 3 i kopirati u sve lokacije odredišnog bloka memorije. Odredišni blokovi počinju od adrese 3000_{16} i nalaze se jedan iza drugoga. Svaki odredišni blok treba sadržavati 20_{16} 32-bitnih podataka. Ako se sa PIO1 učita broj FF_{16}, potrebno je zaustaviti brojenje CT-a te zaustaviti procesor.

Kopiranje vrijednosti u odredišni blok obavlja se pomoću sklopa DMA, koji radi u uvjetnom načinu rada, krađom ciklusa.

Svakih 10 sekundi na sklop PIO2 (koji radi u bezuvjetnom načinu rada) potrebno je poslati ukupni broj pokretanja sklopa DMA. Kašnjenje ostvariti CT-om (spojen na INT2).

4. *(7 bodova*) Za procesor ARM napišite potprogram SAVRSEN koji provjerava je li broj savršen – jednak zbroju svojih pravih djelitelja (tj. zbroju svih svojih djelitelja, uključujući i 1, osim samoga sebe). Na primjer, 28 je savršen jer vrijedi 28=1+2+4+7+14, a 8 nije savršen jer vrijedi 8≠1+2+4. Parametar se u potprogram šalje stogom (uklanja ga pozivatelj), a rezultat se vraća pomoću R0. Rezultat iznosi 1 ako je broj savršen, a 0 ako broj nije savršen.

U glavnom programu treba provjeriti 32-bitne NBC-brojeve iz bloka memorije na adresi 500_{16} i zamijeniti nulom one koji nisu savršeni. U bloku je 10_{10} podataka.

Potprogram SAVRSEN mora djeljivost brojeva provjeravati pomoću potprograma DJELJIVO. Napišite potprogram DJELJIVO koji provjerava je li prvi parametar (prenosi se registrom R1) djeljiv s drugim parametrom (prenosi se lokacijom iza naredbe poziva potprograma). Ako je djeljiv, preko R0 se vraća broj 1, a inače se vraća 0.

5. (*6 bodova*) Računalni sustav inkubatora za patkice sastoji se od procesora ARM te sklopova RTC (spojen na signal FIQ) i GPIO (na koji je spojen termometar opisan na predavanjima).

- vrata B, bitovi [5:0]: iznos temperature, 6-bitni NBC.
- vrata B, bit 6: signal (aktivan visoko) kojim termometar dojavljuje GPIO-u da je postavljena nova temperatura
  vrata B, bit 7: signal (aktivan visoko) kojim GPIO dojavljuje termometru da je pročitao temperaturu
- vrata A, bit 0: izlazni bit kojim se uključuje grijalica (1 uključuje, a 0 isključuje)
- vrata A, bit 1: izlazni bit kojim se uključuje hladilica (1 uključuje, a 0 isključuje)

Napišite program koji upravlja radom inkubatora i treba održavati željenu temperaturu. Temperaturu treba regulirati svakih 60 sekundi (na ulaz RTC-a spojen je signal od 1 Hz). Na početku željena temperatura treba biti 40 stupnjeva. Svaki treći dan (3 dana = 4320 minuta) treba smanjivati željenu temperaturu za 1 stupanj. Nakon 30 dana, treba zaustaviti rad programa.

Završni ispit iz Arhitekture računala 1

15. lipnja 2012. **Grupa na predavanjima: _____**

Prezime i ime (tiskanim slovima): _____ JMBAG: _____

Izjavljujem da tijekom izrade ove zadaće neću od drugoga primiti niti drugome pružiti pomoć, te da se neću koristiti nedopuštenim sredstvima. Ove su radnje teška povreda Kodeksa ponašanja te mogu uzrokovati i trajno isključenje s Fakulteta. Izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Potpis: _____.

Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Završni ispit traje 150 minuta.

1.a. FRISC (1 bod) Koji će biti sadržaj registra R0, nakon izvođenja naredbe LOAD na FRISC-u za programski odsječak s desne strane: _____**00560034**_____

```
ORG 130
 DH 34, 56
 DW 8558
 ...
 LOAD  R0, (130)
```

1.b. FRISC (2,5 boda) Prilikom dekodiranja i izvođenja FRISC-ove naredbe **SUB R1, %D10, R2**, broj 10 se nalazi u 20 nižih bitova registra **____IR_____**. Naredba koristi dva adresiranja koja se zovu **__registarsko__** i ____**neposredno/immediate**_____. Prilikom izvođenja naredbe, na jedan ulaz ALU dovodi se podatak iz sklopa ____**EXT____(nije bitan redoslijed EXT i R1)__,** a na drugi se ulaz dovodi podatak iz __**R1**___ (*napisati iz kojih dijelova procesora se dovodi podatak*).

1.c. FRISC (2 boda) Prekidni sustav procesora FRISC sastoji se od priključaka: ____**INT0-INT1 (ili INT i NMI)**____. Obzirom na mogućnost programske zabrane/dozvoljavanja prekida, procesor FRISC podržava __**maskirajuće**__ prekide na priključku/priključcima __**INT0 (ili INT)**__, i ___**nemaskirajuće** ___ prekide na priključku/priključcima __**INT1 (ili NMI)**___. U registru SR nalazi se prekidna zastavica: ___ **GIE**___. Prilikom prihvaćanja prekida, procesor FRISC automatski sprema povratnu adresu (*gdje*): ____**na stog**__, a stanje registra SR sprema (*gdje*): ___**ne sprema se nigdje**__.

1.d. FRISC (2 boda) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-GPIO zovu se: **____READY ____ i ____ STROBE___.** Ovi priključci koriste se u načinima rada (*nabrojite kojim*): ___**ulazni, izlazni___.** FRISC-GPIO **ne može** postati spreman u načinu/načinima rada: _____**postavljanja bitova**____. ~~Osim registra maske, na prvoj adresi sklopa GPIO nalaze se upravljački registri: _____. Kad šaljemo upravljačku riječ na tu adresu, GPIO zna u koji registar je želimo upisati na temelju (*čega?*): _____~~(ovo pitanje nije primjenjivo na novo gradivo) Ako želimo da GPIO (spojen na adresi FFFF4000) generira maskirajući prekid kad se na bilo kojem od 3 najniža bita postave nule, onda ga inicijaliziramo tako da pošaljemo podatak ___**11100000111**__ na adresu _ **FFFF4000**.

2.a. ARM (**0,5 boda)** Procesor ARM izvodi odsječak programa s desne strane. Nakon izvođenja naredbe LDRB sadržaj registra R1 će biti: ___**0x11**___, a sadržaj registra R0: ___**0x104**___.

```
ORG 0
 MOV R0, #1<8
 LDRB R1, [R0,#4]!
ORG 100
 DW 88776655, 44332211
```

2.b. ARM (0,5 boda) Procesor ARM izvodi naredbu LDMFD R13!, {R2,R1}. Ako je sadržaj registra R13 prije izvođenja naredbe bio 100_{10}, sadržaj registra R13 nakon naredbe će biti __**108_{10}**__ , a registar R1 će se napuniti podatkom s memorijske lokacije __**100_{10}**__.

2.c. ARM (1 bod) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: ___**obradu podataka / AL naredbe**__, __**prijenos podataka / load-store / memorijske**___ i ____**grananje / upravljačke**___. Uvjetno izvođenje kod procesora ARM moguće je za (*koje?*) ___ **sve / gotovo sve**___ naredbe.

2.d. ARM (**0,5 boda**) Za procesor ARM kod naredaba skokova moguća je pojava _____**upravljačkog**_____ hazarda. Negativni efekti ovog hazarda se u nekim procesorima umanjuju pomoću postupka ___**predviđanja grananja**_____ .

2.e. ARM (**1 bod**) Procesor ARM9 uvodi __**harvardsku**____ arhitekturu memorijskog pristupa. Time se izbjegava __**strukturni**___ hazard koji postoji kod ARM7. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave __**podatkovnog**__ hazarda čiji se negativni efekti umanjuju upotrebom ___**prosljeđivanja rezultata/result forwarding**__.

2.f. ARM (**1 bod)** Nakon uključenja procesor ARM 7 treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:_____**16**____. (*napisati postupak rješavanja!*)

```
   ORG 0
    MOV R1, #5                          2c+1c
    ADD R0,R1,#2                        1c
    BIC R0, R0, #%b100      R0=3        1c
A   SUBS  R0,R0,#1                      3 x 1c
    BNE A                           2 x 3c  +  1 x 1c
    EOR R0,R0,R0                        1c
```

2.g. ARM (*2 boda*) Kod ARM-a imamo prekide **_IRQ (ili obični prekid)**_(A) i ___**FIQ (ili brzi prekid)**___(B). Adresa prekidnog potprograma (A) je __**18_{16}**__, a za prekid (B) adresa potprograma je __**$1C_{16}$**___. Za prekid (A) povratna adresa se sprema (*gdje*): __**u LR_irq**__, a za prekid (B) povratna adresa se sprema (*gdje*): __**u LR_fiq**__. Povratak iz prekidnog potprograma (A) ostvaruje se naredbom __**SUBS,PC,LR,#4**__, a iz (B) pomoću naredbe __**SUBS,PC,LR,#4**__.

2.h. ARM (1 bod) Neka priručna memorija ima 16_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
10	**0**
11	**1**
25	**5**
3F	**F**

3. (*7 bodova*) U računalnom sustavu nalaze se procesor FRISC, DMA, CT (na ulaz CNT spojen je signal frekvencije 1 kHz) te dva sklopa GPIO. Adrese vanjskih jedinica odaberite sami.

Procesor u glavnom programu treba neprestano čitati 8-bitne NBC podatke sa sklopa GPIO1 (koji radi u uvjetnom načinu rada). Svaki podatak treba pomnožiti sa 3 i kopirati u sve lokacije odredišnog bloka memorije. Odredišni blokovi počinju od adrese 3000_{16} i nalaze se jedan iza drugoga. Svaki odredišni blok treba sadržavati 20_{16} 32-bitnih podataka. Ako se sa GPIO1 učita broj FF_{16}, potrebno je zaustaviti brojenje CT-a te zaustaviti procesor.

Kopiranje vrijednosti u odredišni blok obavlja se pomoću sklopa DMA, koji radi u uvjetnom načinu rada, krađom ciklusa.

Svakih 10 sekundi na sklop GPIO2 (koji radi u bezuvjetnom načinu rada) potrebno je poslati ukupni broj pokretanja sklopa DMA. Kašnjenje ostvariti CT-om (spojen na INT).

```
PIO1_CR    EQU 0FFFF1000
PIO1_DR    EQU 0FFFF1004
PIO1_STAT  EQU 0FFFF1008
PIO1_END   EQU 0FFFF100C

PIO2_CR    EQU 0FFFF2000
PIO2_DR    EQU 0FFFF2004

DMA_SRC    EQU 0FFFF3000
DMA_DEST   EQU 0FFFF3004
DMA_CNT    EQU 0FFFF3008
DMA_CR     EQU 0FFFF300C
DMA_START  EQU 0FFFF3010
DMA_STAT   EQU 0FFFF3014

CT_CR      EQU 0FFFF4000
CT_LR      EQU 0FFFF4004
CT_IACK    EQU 0FFFF4008
CT_IEND    EQU 0FFFF400C
```

```
ORG 0
     MOVE 10000, SP            ; inicijalizacija stoga
     JP MAIN                   ; skok u glavni program
ORG 8
     DW 2000                   ; prekidni vektor, običan prekid
```

```
GLAVNI
     MOVE 3000, R1              ; R1 - adresa odredišnog bloka za DMA

     MOVE %D 10000, R0          ; 10 kHz * 10 sekundi, CT-konstanta
     STORE R0, (CT_LR)
     MOVE %B11, R0              ; CR, brojilo broji, prekid
     STORE R0, (CT_CR)

     MOVE %B 10000, SR          ; omogućavanje prekida

     MOVE %B 0001, R0           ; PIO1 - ulazni, bez prekida
     STORE R0, (PIO1_CR)

     MOVE %B 010, R0            ; PIO2 - postav. bitova, bez prekida
     STORE R0, (PIO2_CR)

 PETLJA LOAD R0, (PIO1_STAT)          ; čekanje na spremnost PIO1
     CMP R0, 0
     JP_EQ PETLJA
     LOAD R0, (PIO1_DR)         ; učitavanje podatka s PIO1
     CMP R0, 0FF                ; ako je podatak FF -> kraj
     JP_EQ KRAJ

     SHL R0, 1, R2              ; množenje s 3 = množenje s 2 +...
     ADD R2, R0, R0             ; ...+ zbrajanje
     STORE R0, (POMOC)          ; broj treba spremiti u memoriju zbog DMA
     STORE R0, (PIO1_STAT)      ; brisanje spremnosti PIO1
     STORE R0, (PIO1_END)       ; dojava kraja posluživanja PIO1

     MOVE POMOC, R0             ; adresu POMOC postaviti kao izvor DMA
     STORE R0, (DMA_SRC)
     STORE R1, (DMA_DEST)       ; adresa trenutnog odredišnog bloka
     ; Update adrese odredišnog bloka (gornja naredba) nije potreban, jer će
     ; adresni registar u DMA-sklopu tijekom DMA-prijenosa biti točno povećan
     MOVE 20, R0                ; brojač podataka u bloku
     STORE R0, (DMA_CNT)
     MOVE %B 0110, R0           ; kopiranje! SRC-vanjska, DST-memorija
     STORE R0, (DMA_CR)         ; krađa ciklusa, bez prekida
     STORE R0, (DMA_START)      ; pokretanje DMA
DMAW LOAD R0, (DMA_STAT)        ; čekanje na spremnost DMA
     CMP R0,0
     JP_EQ DMAW

     LOAD R0, (BROJ)            ; brojač DMA-prijenosa
     ADD R0, 1, R0
     STORE R0, (BROJ)

     STORE R0, (DMA_STAT)       ; brisanje DMA-status bistabila
     ADD R1, 80, R1             ; povećavanje odredišnog bloka za 4*20=80(16)

     JP PETLJA                  ; skok na novo čekanje
KRAJ MOVE 0, R0                 ; zaustavljanje CT-a
     STORE R0, (CT_CR)
     HALT


     ORG 2000                   ; prekidni potprogram
INT  PUSH R0                    ; čuvanje konteksta
     MOVE SR, R0
     PUSH R0
```

```
      STORE R0, (CT_IACK)        ; potvrda prihvata prekida
      LOAD R0, (BROJ)            ; učitavanje broja pokretanja DMA
      STORE R0, (PIO2_DR)        ; slanje broja na PIO2, bezuvjetno!
      STORE R0, (CT_IEND)        ; kraj posluživanja

      POP R0                     ; obnova konteksta
      MOVE R0, SR
      POP R0
      RETI                       ; povratak iz p.p.

BROJAC DW 0                      ; BROJAC - broj odrađenih DMA-prijenosa
POMOC DW 0                       ; POMOC - mjesto za spremanje broja u mem
```

4. *(7 bodova*) Za procesor ARM napišite potprogram SAVRSEN koji provjerava je li broj savršen – jednak zbroju svojih pravih djelitelja (tj. zbroju svih svojih djelitelja, uključujući i 1, osim samoga sebe). Na primjer, 28 je savršen jer vrijedi 28=1+2+4+7+14, a 8 nije savršen jer vrijedi 8≠1+2+4. Parametar se u potprogram šalje stogom (uklanja ga pozivatelj), a rezultat se vraća pomoću R0. Rezultat iznosi 1 ako je broj savršen, a 0 ako broj nije savršen.

U glavnom programu treba provjeriti 32-bitne NBC-brojeve iz bloka memorije na adresi 500_{16} i zamijeniti nulom one koji nisu savršeni. U bloku je 10_{10} podataka.

Potprogram SAVRSEN mora djeljivost brojeva provjeravati pomoću potprograma DJELJIVO. Napišite potprogram DJELJIVO koji provjerava je li prvi parametar (prenosi se registrom R1) djeljiv s drugim parametrom (prenosi se lokacijom iza naredbe poziva potprograma). Ako je djeljiv, preko R0 se vraća broj 1, a inače se vraća 0.

```
GL    MOV SP, #1<16              ; inicijalizacija stoga
      MOV R6, #5<8               ; R6 – adresa početka bloka (500)
      MOV R2, #0A                ; R2 – broj brojeva u bloku (10)

POC   LDR R0, [R6]               ; učitavanje podatka
      STMFD R13!, {R0}           ; stavljanje parametra na stog
      BL SAVRSEN                 ; poziv potprograma SAVRSEN
      ADD R13, R13, #4           ; brisanje parametra sa stoga
      CMP R0, #0                 ; je li broj NEsavršen?
      STREQ R0, [R6]             ; upisivanje 0 umjesto nesavršenog broja
      ADD R6, R6, #4             ; sljedeći podatak
      SUBS R2, R2, #1            ; smanjivanje broja podataka
      BNE POC                    ; natrag na početak petlje
KRAJ  SWI 123456


SAVRSEN
      STMFD R13!, {R1,R3,R4,R5,R14}   ; kontekst, R14 – gniježđenje
      ADD R3, R13, #14           ; „preskakanje“ konteksta, dolazak do parametra
      LDMFD R1, {R3}             ; R3 – parametar (broj) ili LDR R1,[R13,#14]
      MOV R4, #0                 ; R4 – zbroj djelitelja
      MOV R5, #0                 ; R5 – mogući djelitelji, od 1 do R3
TESTIRAJ
      ADD R5, R5, #1             ; uvećavanje djelitelja
      CMP R1, R5                 ; ako je djelitelj = broj, onda je kraj petlje
      BEQ  GOTOVO

      STR R5, PARAMETAR          ; parametar ide na adresu iz poziva potp.
      BL   DJELJIVO              ; poziv potprograma
PARAMETAR  DW 0                  ; parametar je odmah iza potprograma
      CMP  R0, #1                ; R0 – rezultat. Djeljivost? 1 – da
      ADDEQ R4,R4,R5             ; djeljivo – dodavanje djelitelj u zbroj
      B TESTIRAJ                 ; ponovi petlju za sljedećeg djelitelja

GOTOVO
      CMP R1, R4                 ; usporedba zbroj = početni broj?
      MOVNE R0, #0               ; vraćanje 0 – NEsavršen
      MOVEQ R0, #1               ; vraćanje 1 – savršen
      LDMFD R13!, {R1,R3,R4,R5,R14}; vraćanje konteksta, R14!!!
      MOV PC, LR                 ; povratak iz potprograma SAVRSEN


DJELJIVO
      STMFD R13!, {R1,R2}        ; spremanje konteksta
      MOV R0, #0                 ; priprema povratne vrijednosti false (0)
                                 ; R1 – prvi parametar
      LDR R2, [LR],#4            ; R2 – drugi parametar i istovremeno
                                 ; povećavanje LR (može i posebnom naredbom)
PONOVO SUBS R1,R1,R2             ; uzastopno oduzimanje sve dok je...
      BHI PONOVO                 ; ..."dijeljenik" veći od djelitelja
```

```
      MOVEQ R0, #1              ; vraćanje true (1) ako je djeljiv

      LDMFD R13!, {R1,R2}
      MOV PC, LR
ORG 500
      DW 1,2,8,%D28,5,6,7,8,9,0A
```

5. (*6 bodova*) Računalni sustav inkubatora za patkice sastoji se od procesora ARM te sklopova RTC (spojen na signal FIQ) i GPIO (na koji je spojen termometar opisan na predavanjima).

- vrata B, bitovi [5:0]: iznos temperature, 6-bitni NBC.
- vrata B, bit 6: signal (aktivan visoko) kojim termometar dojavljuje GPIO-u da je postavljena nova temperatura
  vrata B, bit 7: signal (aktivan visoko) kojim GPIO dojavljuje termometru da je pročitao temperaturu
- vrata A, bit 0: izlazni bit kojim se uključuje grijalica (1 uključuje, a 0 isključuje)
- vrata A, bit 1: izlazni bit kojim se uključuje hladilica (1 uključuje, a 0 isključuje)

Napišite program koji upravlja radom inkubatora i treba održavati željenu temperaturu. Temperaturu treba regulirati svakih 60 sekundi (na ulaz RTC-a spojen je signal od 1 Hz). Na početku željena temperatura treba biti 40 stupnjeva. Svaki treći dan (3 dana = 4320 minuta) treba smanjivati željenu temperaturu za 1 stupanj. Nakon 30 dana, treba zaustaviti rad programa.

```
      ORG 0
      B GLAVNI                  ; skok na glavni program

      ORG 1C                    ; prekidni potprogram – FIQ

      STMFD R13!,{R0,R1,R4,R5,R6,R7} ; spremanje konteksta za opće registre
      LDR R0, GPIO              ; pristojno je ponovno učitati bazne adrese,
      LDR R1, RTC               ; a ne prenositi ih registrima u p.p.

      MOV R9, #0                ;
      STR R9, [R1,#0C]          ; resetiranje brojača
      STR R9, [R1,#8]           ; dojava prihvata prekida

ČITAJ LDR R10, [R0,#4]          ; čitanje podatka s vrata B
      ANDS R12, R10, #40        ; 40 = %B 100 0000; bit 6 je signal nove temp.
      BEQ ČITAJ                 ; čekanje na novu temperaturu

      MOV R12, #80              ; 80 = %B 1000 0000; bit 7 u visoko
      STR R12, [R0,#4]          ; slanje na vrata B
      MOV R12, #00              ; 80 = %B 0000 0000; bit 7 natrag u nisko
      STR R12, [R0,#4]          ; slanje na vrata B

      AND R10, R10, #3F         ; 3F = 0011 1111; čitanje temperature [5:0]
      LDR R4, ZELJENA           ; učitavanje željene temperature iz mem
      CMP R10, R4               ; usporedba sa željenom temperaturom
      MOVHI R10, #2             ; veća temp – hladilica! = 10
      MOVMI R10, #1             ; manja temp – grijalica! = 01
      MOVEQ R10, #0             ; ista temperatura – isključiti sve = 00
      STR R10, [R0]             ; spremanje na vrata A

      LDR R5, MINUTEU3          ; učitavanje trenutno proteklih minuta
      ADD R5, R5, #1            ; prošla je jedna minuta (zatražen prekid)
      STR R5, MINUTEU3          ; spremanje trenutno proteklih minuta

      LDR R7, MIN4320           ; učitavanje broja minuta u 3 dana
      CMP R5, R7                ; jesu li prošla 3 dana?
      BNE  VAN                  ; ako nisu prosla 3 dana -> preskoči
```

```
JESU3 SUB R4, R4, #1            ; prošla 3 dana -> smanjiti željenu temp. za 1
      STR R4, ZELJENA           ; spremanje željene temperature u mem
      MOV R5, #0                ; vratiti brojač minuta na 0
      STR R5, MINUTEU3

      LDR R6, TRI_DANA          ; učitavanje i povećanje brojača trenutno...
      ADD R6, R6, #1            ; ...proteklih trodnevnih razdoblja
      STR R6, TRI_DANA

      CMP R6, #%D 10            ; je li proteklo 10 trodnevlja (tj. 30 dana)?
      BNE  VAN                  ; ako nije prošlo 30 dana -> izađi iz prekida
      SWI 123456                ; inače je kraj programa

VAN   LDMFD R13!,{R0,R1,R4,R5,R6,R7} ; vraćanje konteksta
      SUBS PC, LR, #4           ; povratak iz prekidnog potprograma

GLAVNI
      MOV R13,#10<12            ; inicijalizacija stoga
      LDR R0, GPIO              ; bazna adresa sklopa GPIO
      LDR R1, RTC               ; bazna adresa sklopa RTC

      MOV R2, #3                ; 3 = %B 11, registar smjera vrata A
      STR R2, [R0,#8]           ; bitovi 0 i 1 su izlazni (1)
      MOV R2, #7F               ; 7F = %B 0111 1111, registar smjera vrata B
      STR R2, [R0,#C]           ; bit 7 je izlazni (0), ostali ulazni (1)
      MOV R2, #1                ; 1 - prekidi
      STR R2, [R1,#10]          ; upis u upravljački registar (CTCR)
      MOV R2, #%D60             ; 60 sekundi
      STR R2, [R1,#4]           ; - upis u RTCMR

      MRS R2, CPSR              ; omogućavanje prekida FIQ
      BIC R2, R2, #40
      MSR CPSR_c, R2
PETLJA     B PETLJA             ; prazna petlja


          ORG 300
GPIO      DW FFFF1000
RTC       DW FFFF2000

MINUTEU3  DW 0                  ; trenutno prošle minute u 3-dnevnom periodu
TRI_DANA  DW 0                  ; trenutno protekli 3-dnevni periodi
ZELJENA   DW %D 40              ; željena temperatura, na početku 40
MIN4320   DW %D 4320            ; broj minuta u 3 dana
```

Komentar: bilo bi "ljepše" da se procesor zaustavlja unutar glavnog programa, a ne kao u ovom rješenju gdje se to radi u prekidnomu. Na primjer, pomoću neke zastavice/varijable koju glavni program ispituje, a postavlja je prekidni program nakon isteka 30 dana.