Završni ispit iz Arhitekture računala 1 — 18. lipnja 2015.

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 ovog ispita. Potpis: _____.

Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Rješenja teorijskih zadatka treba napisati na ovaj papir. Završni ispit traje 135 minuta.

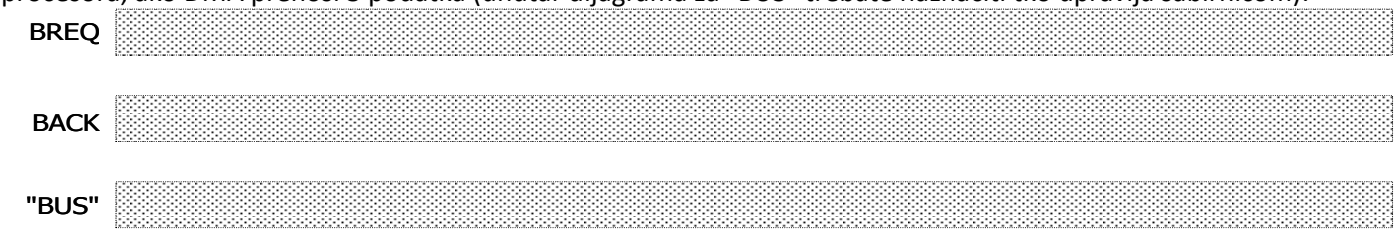
1.a (1 bod): 4-bitna ALU oduzima binarne brojeve 1101-1010. Napišite stanja zastavica poslije oduzimanja. Prijenos=___, Posudba=___, Preljev=___, Ništica=___, Predznak=___. **Mora se vidjeti način izračunavanja zastavica.**

1. b. (1 bod): Podatak 101011_2 u 6-bitnom NBC-u predstavlja broj _____, a u 6-bitnom formatu 2'k predstavlja broj _____. Podatak 0101_2 u 4-bitnom NBC-u predstavlja broj _____, a u 4-bitnom formatu 2'k predstavlja _____.

2.a (1 bod): FRISC-GPIO za spajanje s vanjskim uređajima ima 8 priključaka za podatke, priključak _____ čiji smjer je _____ i priključak _____ čiji smjer je _____. Ova dva priključka nazivaju se _____.

2.b (2 boda): Sklop FRISC-DMA treba kopirati podatak sa memorijske lokacije 1000 u sve lokacije memorijskog bloka od adrese 2000 do adrese 20FC. Kopiranje treba obaviti pomoću krađe ciklusa, a kraj prijenosa javiti prekidom. DMA je na adresi FFFF0000. Napišite koje sve podatke pri inicijalizaciji treba poslati DMA-sklopu i na koje adrese:
podatak _____ se šalje na adresu_____
podatak _____ se šalje na adresu_____
podatak _____ se šalje na adresu_____
podatak _____ se šalje na adresu_____
Nakon toga, DMA-prijenos treba pokrenuti slanjem podatka _____ na adresu _____.

2.c (2 boda) Nacrtajte vremenski dijagram stanja na sabirničkim linijama prilikom DMA-prijenosa zaustavljanjem procesora, ako DMA prenosi 3 podatka (unutar dijagrama za "BUS" trebate naznačiti tko upravlja sabirnicom):



3.a (2 boda): Koliko traje izvođenje ovog odsječka na procesoru ARM7: ____ciklusa. Kraj svake naredbe napišite koliko puta se izvodi i koliko traje pojedino izvođenje (npr. 3 x 1c + 1 x 2c znači da naredba tri puta traje 1 ciklus i jednom traje 2 ciklusa).

```
      MOV R0,#3          _____
      MOV R1,#0          _____
LOOP  LDR R2,[R10],#4    _____
      ADD R1,R1,R2       _____
      SUBS R0,R0,#1      _____
      BNE LOOP           _____
      STR R1,[R10]       _____
```

3.b (0,5 boda): Početno je R13 = 100_{10}, a nakon **`STMIA R13!,{R1,R2,R3}`** u R13 bit će vrijednost _____. Ako u naredbi **`STMIA`** umjesto "IA" želimo koristiti nastavak za rad sa stogom, naredba će glasiti:_____.

3.c (1 bod): ARM na sabirnici AMBA-AHB pristupa brzoj memoriji. Pristup se sastoji od _____ i _____ faze, od kojih svaka traje ____ takt(ova) CLOCK-a. Cijeli pristup efektivno traje samo _____ takt(ova) jer _____
_____(objasnite u 1 rečenici, što ARM radi s fazama).

3.d (1 bod): Prilikom statičkog predviđanja skoka, uspoređuju se dvije vrijednosti: _____ i _____. Ako je skok prema "natrag", tada se predviđa da će se _____. Predviđanjem skoka, pokušava se umanjiti utjecaj _____ hazarda.

3.e (1,5 bod): Prilikom prihvata brzog prekida FIQ, ARM automatski izvodi sljedeće korake:

_____
_____
_____
_____
_____
_____
_____

4. FRISC (10 bodova) Na FRISC su spojena dva sklopa GPIO i jedan CT. Adrese vanjskih jedinica odaberite sami.

Napisati program za FRISC koji pomoću sklopa GPIO_1 prima podatke i sprema ih kao 8-bitne podatke u memoriju od adrese 1000_{16}. Spremnost sklopa GPIO_1 treba ispitivati programski (tj. uvjetno).

Kad se sa GPIO_1 primi 100_{16} podataka, treba prestati s primanjem pa pričekati 5 sekundi. Kašnjenje treba ostvariti pomoću sklopa CT koji postavlja zahtjev za prekid NMI. Na CT-ov priključak CNT spojen je signal frekvencije 1 kHZ.

Nakon što protekne 5 sekundi, treba bezuvjetno poslati 100_{16} 8-bitnih podataka sa adrese 1000_{16} na sklop GPIO_2, ali tako da se između svaka dva slanja čeka 0,1 sekundu (i ovo kašnjenje treba ostvariti pomoću CT-a i prekida NMI). Nakon toga treba zaustaviti procesor.

5. ARM (7 bodova) Za procesor ARM napišite potprogram TOUPPER koji prima jedan ASCII-znak preko stoga. Znak treba ispitati i ako se radi o malom slovu (znakovi 'a' do 'z' imaju ASCII-kodove od 97_{10} do 122_{10}), onda ga treba pretvoriti u veliko slovo (znakovi 'A' do 'Z' imaju ASCII-kodove od 65_{10} do 90_{10}). Ako se radi o ostalim ASCII-znakovima, oni ostaju nepromijenjeni. Povratni ASCII-znak vraća se pozivatelju preko registra R0.

Napišite potprogram POTP koji treba sva mala slova u stringu pretvoriti u velika, i to korištenjem potprograma TOUPPER. POTP prima adresu stringa preko lokacije iza naredbe poziva potprograma. String je niz bajtova u kojima su ASCII-znakovi terminirani NUL-znakom (ASCII kod 0).

Napišite glavni program koji obrađuje tekst smješten u memoriji tako da u cijelom tekstu pomoću potprograma POTP pretvara sva mala slova u velika. Tekst je u memoriji smješten kao niz stringova i to na sljedeći način. Stringovi se nalaze na memorijskim adresama koje nisu fiksno zadane, ali sve adrese stringova zapisane su redom od adrese 1000_{16} na dalje. Broj stringova također nije fiksno zadan, ali je niz njihovih adresa terminiran podatkom FFFFFFFF.

6. ARM (10 bodova) ARM sa GPIO-om i RTC-om (adrese im odaberite sami) beskonačno ispituje temperaturu i na temelju njene razlike od željene temperature upravlja uređajem za grijanje/hlađenje. Korisnik može u bilo kojem trenutku promijeniti željenu temperaturu.

Na vrata A sklopa GPIO spojen je temperaturni sklop kao na predavanjima:
- bitovi 0-5, ulazni, iznos temperature
- bit 6, ulazni, potvrda od temperaturnog sklopa da je postavljena valjana temperatura
- bit 7, izlazni, dojava temperaturnom sklopu da je temperatura pročitana

Na vrata B sklopa GPIO spojene su dvije tipke $\oplus$ i $\ominus$. Također je spojen uređaj za grijanje/hlađenje:
- bit 0, ulazni, spojena je tipka $\oplus$ za povećavanje željene temperature za 1 stupanj
- bit 1, ulazni, spojena je tipka $\ominus$ za smanjivanje željene temperature za 1 stupanj
- bitovi 2-3, izlazni, spojen je uređaj za grijanje/hlađenje (00=uređaj je isključen, 01=uređaj grije, 11=uređaj hladi)

Svake sekunde treba očitati temperaturu i na temelju toga upravljati uređajem za grijanje/hlađenje. Kašnjenje treba ostvariti sklopom RTC na koji je spojen 1 kHZ, a RTC generira IRQ.

Početna željena temperatura je 20 stupnjeva. Tipke $\oplus$ i $\ominus$ (na bitovima 0 i 1 na portu B) postavljaju stanje 1 kad su pritisnute, a 0 kad nisu pritisnute. Pomoću njih korisnik može uvećati ili umanjiti željenu temperaturu za 1 stupanj, ali ne izvan granica od 10 do 30 stupnjeva (takvi pokušaji se ignoriraju). Da bi se utvrdilo da je tipka pritisnuta, prvo treba čekati da odgovarajući priključak na portu B postane 1, a zatim da se tipka otpusti, tj. da postane 0. Zbog jednostavnosti, pretpostavite da korisnik nikad ne pritisne obje tipke istodobno. Također pretpostavite da se tipke ignoriraju tijekom obrade prekida.

RJEŠENJA

1.a (1 bod): 4-bitna ALU oduzima binarne brojeve 1101-1010. Napišite stanja zastavica poslije oduzimanja. Prijenos=_1_, Posudba=_0_, Preljev=_0_, Ništica=_0_, Predznak=_0_. **Mora se vidjeti način izračunavanja zastavica.**

1. b. (1 bod): Podatak 101011_2 u 6-bitnom NBC-u predstavlja broj ___43___, a u 6-bitnom formatu 2'k predstavlja broj __-21_. Podatak 0101_2 u 4-bitnom NBC-u predstavlja broj __5___, a u 4-bitnom formatu 2'k predstavlja __5___.

2.a (1 bod): FRISC-GPIO za spajanje s vanjskim uređajima ima 8 priključaka za podatke, priključak ___READY___ čiji smjer je ___izlazni_____ i priključak ___STROBE_____ čiji smjer je ___ulazni____. Ova dva priključka nazivaju se __priključcima za rukovanje (ili sinkronizaciju ili handshaking)_____.

2.b (2 boda): Sklop FRISC-DMA treba kopirati podatak sa memorijske lokacije 1000 u sve lokacije memorijskog bloka od adrese 2000 do adrese 20FC. Kopiranje treba obaviti pomoću krađe ciklusa, a kraj prijenosa javiti prekidom. DMA je na adresi FFFF0000. Napišite koje sve podatke pri inicijalizaciji treba poslati DMA-sklopu i na koje adrese:
podatak _____1000_____ se šalje na adresu_____FFFF0000_____
podatak _____2000_____ se šalje na adresu_____FFFF0004_____
podatak _40 ili 3F (64 ili 63_{10})_ se šalje na adresu_____FFFF0008_____
podatak ___%B 0111_____ se šalje na adresu_____FFFF000C_____
Nakon toga, DMA-prijenos treba pokrenuti slanjem podatka ___bilo kojeg_____ na adresu __FFFF0010____.

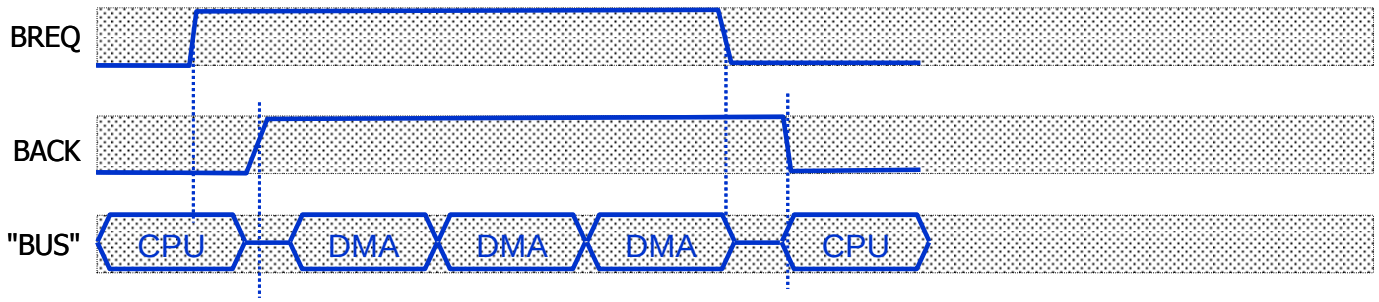
2.c (2 boda) Nacrtajte **vremenski dijagram stanja na sabirničkim linijama** prilikom DMA-prijenosa zaustavljanjem procesora, ako DMA prenosi 3 podatka (unutar dijagrama za "BUS" trebate naznačiti tko upravlja sabirnicom)::

BREQ se aktivira tijekom CPU-ciklusa

BREQ se aktivira na kraju DMA-ciklusa

BACK se aktivira na kraju ili iza CPU-ciklusa

BACK se aktivira na početku CPU-ciklusa



3.a (2 boda): Koliko traje izvođenje ovog odsječka na procesoru ARM7: _28_ ciklusa. Kraj svake naredbe napišite koliko puta se izvodi i koliko traje pojedino izvođenje (npr. 3 x 1c + 1 x 2c znači da naredba tri puta traje 1 ciklus i jednom traje 2 ciklusa).

```
        MOV R0,#3          __1 x 1c + 2c__
        MOV R1,#0          __1 x 1c_____
LOOP    LDR R2,[R10],#4    __3 x 3c_____
        ADD R1,R1,R2       __3 x 1c_____
        SUBS R0,R0,#1      __3 x 1c_____
        BNE LOOP           2 x 3c + 1 x 1c
        STR R1,[R10]       __1 x 2c_____
```

3.b (0,5 boda): Početno je R13 = 100_{10}, a nakon **`STMIA R13!,{R1,R2,R3}`** u R13 bit će vrijednost __112__. Ako u naredbi **`STMIA`** umjesto "IA" želimo koristiti nastavak za rad sa stogom, naredba će glasiti:__STMEA_____.

3.c (1 bod): ARM na sabirnici AMBA-AHB pristupa brzoj memoriji. Pristup se sastoji od __adresne__ i __podatkovne__ faze, od kojih svaka traje _1_takt(ova) CLOCK-a. Cijeli pristup efektivno traje samo __1_takt(ova) jer __se adresna faza trenutnog pristupa preklapa s podatkovnom fazom prethodnog pristupa___ (objasnite u 1 rečenici, što ARM radi s fazama).

3.d (1 bod): Prilikom statičkog predviđanja skoka, uspoređuju se dvije vrijednosti: _PC ili adresa naredbe skoka_i __adresa odredišta skoka_. Ako je skok prema "natrag", tada se predviđa da će se __doći do grananja/skoka_. Predviđanjem skoka, pokušava se umanjiti utjecaj __upravljačkog ili kontrolnog__ hazarda.

3.e (1,5 bod): Prilikom prihvata brzog prekida FIQ, ARM automatski izvodi sljedeće korake:

__R14_fiq = adresa sljedeće naredbe + 4_____

__SPSR_fiq = CPSR_____

__CPSR [0..4.] = 10001 ____ili prelazak u FIQ način rada_____

__CPSR[5] = 0 ____ili prelazak u arm način rada/izvođenja_____

__CPSR[6] = 1 ____ili onemogući/zabrani brze prekide/FIQ_____

__CPSR[7] = 1 ____ili onemogući/zabrani obične prekide/IRQ_____

__PC = 1C_____

4. FRISC (10 bodova) Na FRISC su spojena dva sklopa GPIO i jedan CT. Adrese vanjskih jedinica odaberite sami.

Napisati program za FRISC koji pomoću sklopa GPIO_1 prima podatke i sprema ih kao 8-bitne podatke u memoriju od adrese 1000_{16}. Spremnost sklopa GPIO_1 treba ispitivati programski (tj. uvjetno).

Kad se sa GPIO_1 primi 100_{16} podataka, treba prestati s primanjem pa pričekati 5 sekundi. Kašnjenje treba ostvariti pomoću sklopa CT koji postavlja zahtjev za prekid NMI. Na CT-ov priključak CNT spojen je signal frekvencije 1 kHZ.

Nakon što protekne 5 sekundi, treba bezuvjetno poslati 100_{16} 8-bitnih podataka sa adrese 1000_{16} na sklop GPIO_2, ali tako da se između svaka dva slanja čeka 0,1 sekundu (i ovo kašnjenje treba ostvariti pomoću CT-a i prekida NMI). Nakon toga treba zaustaviti procesor.

```
PIO1_CR     EQU   0FFFF0000     ; adrese svih VJ
PIO1_DR     EQU   0FFFF0004
PIO1_STAT   EQU   0FFFF0008
PIO1_END    EQU   0FFFF000C

PIO2_CR     EQU   0FFFF1000
PIO2_DR     EQU   0FFFF1004

CT_CR       EQU   0FFFF1000
CT_LR       EQU   0FFFF1004
CT_IACK     EQU   0FFFF1008
CT_IEND     EQU   0FFFF100C

        ORG   0
        MOVE  10000, SP
        JP    GLAVNI

;*****************************************************************
        ORG   0C      ; potprogram za NMI

        PUSH  R0      ; spremanje konteksta
        PUSH  R1
        MOVE  SR, R0
        PUSH  R0

        STORE R0, (CT_IACK)   ; dojava prihvata prekida

        LOAD  R0, (BROJAC)    ; provjera stanja slanja podataka:
        CMP   R0, 100         ; ako je brojač = 100, onda još nije bilo slanja i treba poslati prvi podatak
        JR_NE U_SLANJU        ; ako je brojač != 100, onda je slanje u tijeku

; prvi prekid: sve je primljeno i 5 sekundi je prošlo
; preprogramiraj CT na 0,1 sekundu, treba poslati samo novu konstantu, a upravljačka riječ i dalje vrijedi
PRVO_SLANJE
        MOVE  %D 100, R0
        STORE R0, (CT_LR)
        ; nastavi dalje s izvođenjem, tj. pošalji prvi podatak

U_SLANJU      ; inače se nalazimo unutar slanja
        LOAD  R1, (ADRESA)    ; dohvati adresu
        LOADB R0, (R1)        ; dohvati podatak (bajt)
        ADD   R1, 1, R1       ; povećaj i spremi adresu
        STORE R1, (ADRESA)

        STORE R0, (PIO2_DR)   ; dohvaćeni podatak bezuvjetno šalji na PIO_2

        LOAD  R0, (BROJAC)    ; dohvati, smanji i spremi brojač poslanih podataka
        SUB   R0, 1, R0
        STORE R0, (BROJAC)

        CMP   R0, 0           ; provjeri je li sve poslano
        JR_NE VAN             ; ako nije sve poslano, onda idi na normalan izlazak
```

```
SVE POSLANO  ; ako je sve poslano, treba postaviti oznaku GOTOVO (za zaustavljanje)
        MOVE  1, R0
        STORE R0, (GOTOVO)                    nije lijepo, ali može i ovdje samo HALT

VAN     STORE R0, (CT_IEND)  ; dojava kraja posluživanja
        POP   R0             ; obnova konteksta i povratak
        MOVE  R0, SR
        POP   R1
        POP   R0
        RETN


GOTOVO      DW    0      ; oznaka za zaustavljanje procesora

ADRESA      DW    1000   ; adresa za slanje na PIO_2
BROJAC      DW    100    ; brojač za slanje na PIO_2


;*************************************************************
GLAVNI; početak glavnog programa
        MOVE  %B 0001, R0    ; inicijalizacija ulaznog načina bez prekida za PIO_1
        STORE R0, (PIO1_CR)

        MOVE  %B 0010, R0    ; inicijalizacija postavljanja bitova za PIO_2 (može i izlazni 0000)
        STORE R0, (PIO2_CR)

        ; petlja za primanje 100 podataka od PIO_1 i njihovo spremanje u memoriju na adresu 1000
        MOVE  1000, R1       ; adresa za spremanje
        MOVE  100, R2        ; brojač za petlju

CEKAJ   LOAD  R0, (PIO1_STAT)        ; čekaj spremnost PIO_1
        CMP   R0, 0
        JR_EQ CEKAJ

        STORE R0, (PIO1_STAT)        ; briši spremnost PIO_1
        LOAD  R0, (PIO1_DR)          ; primi podatak od PIO_1
        STORE R0, (PIO1_END)         ; dojavi kraj posluživanja

        STORE R0, (R1)       ; spremi podatak u memoriju (bajt)
        ADD   R1, 1, R1      ; povećaj adresu

        SUB   R2, 1, R2      ; smanji brojač petlje i ponavljaj petlju
        JR_NZ CEKAJ

SVE_PRIMLJENO        ; svih 100 podataka je primljeno, inicijalizirati CT da proizvede kašnjenje od 5 sekundi
        MOVE  %D 5000, R0    ; vremenska konstanta
        STORE R0, (CT_LR)
        MOVE  %B 111, R0     ; upravljačka riječ: CT radi i generira NMI
        STORE R0, (CT_CR)

LOOP    LOAD  R0, (GOTOVO)   ; prazna petlja s ispitivanjem oznake za zaustavljanje procesora
        CMP   R0, 0
        JR_EQ LOOP

        HALT
```

Napomena: Prekidni potprogram može se napisati kraće, ali uz preprogramiranje CT-a kod svakog prekida. Bilo bi dovoljno provjeriti kraj podataka i poslati podatak ako nije kraj, a zatim preprogramirati CT na 0,1 sek.

5. ARM (7 bodova) Za procesor ARM napišite potprogram TOUPPER koji prima jedan ASCII-znak preko stoga. Znak treba ispitati i ako se radi o malom slovu (znakovi 'a' do 'z' imaju ASCII-kodove od 97_{10} do 122_{10}), onda ga treba pretvoriti u veliko slovo (znakovi 'A' do 'Z' imaju ASCII-kodove od 65_{10} do 90_{10}). Ako se radi o ostalim ASCII-znakovima, oni ostaju nepromijenjeni. Povratni ASCII-znak vraća se pozivatelju preko registra R0.

Napišite potprogram POTP koji treba sva mala slova u stringu pretvoriti u velika, i to korištenjem potprograma TOUPPER. POTP prima adresu stringa preko lokacije iza naredbe poziva potprograma. String je niz bajtova u kojima su ASCII-znakovi terminirani NUL-znakom (ASCII kod 0).

Napišite glavni program koji obrađuje tekst smješten u memoriji tako da u cijelom tekstu pomoću potprograma POTP pretvara sva mala slova u velika. Tekst je u memoriji smješten kao niz stringova i to na sljedeći način. Stringovi se nalaze na memorijskim adresama koje nisu fiksno zadane, ali sve adrese stringova zapisane su redom od adrese 1000_{16} na dalje. Broj stringova također nije fiksno zadan, ali je niz njihovih adresa terminiran podatkom FFFFFFFF.

```
;*****************************************************************

GLAVNI MOVE  SP, #10<12      ; inicijalizacija pokazivača stoga
       MOVE  R2, #10<8       ; na 1000 je niz adresa od stringova

LOOP   LDR   R1, [R2], #4    ; učitavanje adrese jednog stringa s pomakom adrese
       CMN   R1, #1          ; usporedba sa terminatorom FFFFFFFF ...
       BEQ   KRAJ            ; ... i izlazak iz petlje u slučaju pronalaska teminatora

       STR   R1, PARAM       ; stavi adresu na mjesto za parametar
       BL    POTP            ; obradi string pomoću potprograma
PARAM  DW    0               ; mjesto za slanje parametra

       B     LOOP            ; ponovi za sljedeći string

KRAJ   SWI 123456

;*****************************************************************

POTP   STMFD SP!, {R0,R1}    ; spremi kontekst
       LDR   R1, [LR], #4    ; pročitaj parametar u R1 i pomakni adresu povratka
       STMFD SP!,{LR}        ; LR se tek sada sprema jer je prethodno morao biti povećan za 4.
                             ; Alternativni načini spremanja i obnove konteksta su prikazani ispod rješenja.

       ; u R1 je adresa stringa, treba obraditi znak po znak

LOOP1  LDRB  R0, [R1]        ; dohvaćanje pojedinih znakova iz stringa u registar R0
       CMP   R0, #0          ; ispitivanje kraja stringa (usporedba s NUL-znakom, tj. sa nulom)
       BEQ   VAN             ; ako je NUL-znak, onda izlazak izvan petlje

       STMFD SP!, {R0}       ; znak iz R0 treba staviti na stog kao parametar za TOUPPER
       BL    TOUPPER
       ADD   SP, SP, #4      ; ukloniti parametar sa stoga
       STRB  R0, [R1],#1     ; spremi obrađeni znak natrag u string i pomakni adresu na sljedeći znak

       B     LOOP1           ; natrag na obradu sljedećeg znaka u stringu

VAN    LDMFD SP!, {LR}       ; obnova konteksta s povratkom - ne može jedna naredba LDM zbog redoslijeda
                             ; spremanja registara (niži indeks - niža adresa)
       LDMFD SP!,{R0,R1}
       MOV   PC, LR
```

```
;*****************************************************************

TOUPPER     ; nema spremanja konteksta jer se mijenja samo R0 preko kojega se ionako vraća rezultat
        LDR     R0, [SP]        ; čitanje parametra (tj. ASCII-znaka) sa stoga

        CMP     R0, #%D 97      ; usporedba znaka sa slovom 'a'
        BLO     IZLAZ           ; ako je znak "manji" od 'a', nema promjene  (može i MOVLO PC,LR)

        CMP     R0, #%D 122     ; usporedba znaka sa slovom 'z'
        BHI     IZLAZ           ; ako je znak "veći" od 'z', nema promjene  (može i MOVHI PC,LR)

        ; inače je znak između 'a' i 'z' i treba ga pretvoriti u veliko slovo
        SUB     R0, R0, #%D 32 ; pretvorba iz malih u velika slova, rezultat se vraća preko R0

IZLAZ   MOV     PC, LR          ; povratak
```

DRUGAČIJE RJEŠENJE spremanja konteksta za POTP:

```
POTP    STMFD SP!, {R0,R1,LR}
        LDR     R1, [LR]
        ...
VAN     LDMFD SP!, {R0,R1, LR}
        ADD     LR, LR, #4
        MOV     PC, LR
```

ili još kraće:

```
        ...
VAN     LDMFD SP!, {R0,R1, LR}
        ADD     PC, LR, #4
```

6. ARM (10 bodova) ARM sa GPIO-om i RTC-om (adrese im odaberite sami) beskonačno ispituje temperaturu i na temelju njene razlike od željene temperature upravlja uređajem za grijanje/hlađenje. Korisnik može u bilo kojem trenutku promijeniti željenu temperaturu.

Na vrata A sklopa GPIO spojen je temperaturni sklop kao na predavanjima:

- bitovi 0-5, ulazni, iznos temperature
- bit 6, ulazni, potvrda od temperaturnog sklopa da je postavljena valjana temperatura
- bit 7, izlazni, dojava temperaturnom sklopu da je temperatura pročitana

Na vrata B sklopa GPIO spojene su dvije tipke ⊕ i ⊖. Također je spojen uređaj za grijanje/hlađenje:

- bit 0, ulazni, spojena je tipka ⊕ za povećavanje željene temperature za 1 stupanj
- bit 1, ulazni, spojena je tipka ⊖ za smanjivanje željene temperature za 1 stupanj
- bitovi 2-3, izlazni, spojen je uređaj za grijanje/hlađenje (00=uređaj je isključen, 01=uređaj grije, 11=uređaj hladi)

Svake sekunde treba očitati temperaturu i na temelju toga upravljati uređajem za grijanje/hlađenje. Kašnjenje treba ostvariti sklopom RTC na koji je spojen 1 kHZ, a RTC generira IRQ.

Početna željena temperatura je 20 stupnjeva. Tipke ⊕ i ⊖ (na bitovima 0 i 1 na portu B) postavljaju stanje 1 kad su pritisnute, a 0 kad nisu pritisnute. Pomoću njih korisnik može uvećati ili umanjiti željenu temperaturu za 1 stupanj, ali ne izvan granica od 10 do 30 stupnjeva (takvi pokušaji se ignoriraju). Da bi se utvrdilo da je tipka pritisnuta, prvo treba čekati da odgovarajući priključak na portu B postane 1, a zatim da se tipka otpusti, tj. da postane 0. Zbog jednostavnosti, pretpostavite da korisnik nikad ne pritisne obje tipke istodobno. Također pretpostavite da se tipke ignoriraju tijekom obrade prekida.

```
        ORG     0
        B GLAVNI

;*****************************************************************
        ORG 18
PREKIDNI        ; prekidni potprogram za IRQ, FIQ se ne koristi pa se može napisati ovdje
        STMFD SP!, {R0, R1, R2}         ; spremi kontekst

        LDR     R1, GPIO_ADR            ; dohvat adresa od GPIO i CT
        LDR     R2, CT_ADR

        ; posluži CT
        MOV     R0, #0
        STR     R0, [R2, #0C]    ; obriši brojilo
        STR     R0, [R2, #08]    ; dojavi prihvat prekida

        ; očitanje temperature sa vrata A od sklopa GPIO
CEKAJ   LDR     R0, [R1]                ; čekaj na spremnost temperaturnog sklopa, tj. čekaj...
        ANDS    R2, R0, #%B 01000000  ; ... na dizanje u jedinicu bita 6 porta A
        BEQ     CEKAJ

        ORR     R0, R0, #%B 10000000  ; digni i spusti signal na bitu 7 porta A
        STR     R0, [R1]
        EOR     R0, R0, #%B 10000000
        STR     R0, [R1]

        ; upravljanje grijanjem/hlađenjem na temelju očitane temperature
        AND     R0, R0, #%B 00111111  ; čišćenje viška bitova da se dobije iznos temperature u R0
        LDR     R2, ZELJENA             ; dohvat željene temperature
        CMP     R0, R2                  ; usporedi: trenutna <?> željena

        ; upravljanje - tj. postavljanje bitova za uključivanje isključivanje uređaja za grijanje/hlađenje
        MOVEQ       R2,%B 0000      ; ako je temperatura jednaka željenoj, isključi uređaj za grijanje/hlađenje
        MOVHI       R2,%B 1100      ; ako je temperatura veća od željene, uključi hlađenje
        MOVLO       R2,%B 0100      ; ako je temperatura manja od željene, uključi grijanje

        STR     R2, [R1,#4]             ; pošalji stanje uređaja na port B

        LDMFD SP!, {R0, R1, R2}         ; obnova konteksta i izlazak
        SUBS    PC, LR, #4
```

```
;*****************************************************************
GLAVNI MOVE  SP, #10<12          ; inicijalizacija stoga i adresa VJ-a
       LDR   R1, GPIO_ADR
       LDR   R2, CT_ADR

       MOV   R0, #%B 10000000    ; smjerovi na vratima A od GPIO-a
       STR   R0, [R1,#8]

       MOV   R0, #%B 00000011    ; smjerovi na vratima B od GPIO-a
       STR   R0, [R1,#0C]

       LDR   R0, TISUCU          ; konstanta brojenja za CT (može i MOV R0,%D250<2)
       STR   R0, [R2,#4]

       MOV   R0, #%B 1           ; upravljačka riječ za CT (koriste se prekidi)
       STR   R0, [R2,#10]

       MRS   R0, CPSR            ; dozvoli IRQ
       BIC   R0, R0, #%B 10000000
       MSR   CPSR_c, R0

TIPKE  LDR   R0, [R1, #4]        ; učitati i ispitati stanje tipaka sa porta B
       ANDS  R3, R0, #%B 1       ; eliminirati sve ostale bitove osim tipke +
       BNE   PLUS                ; pritisnut je +
       ANDS  R3, R0, #%B 10      ; eliminirati sve ostale bitove osim tipke -
       BNE   MINUS               ; pritisnut je -
       B     TIPKE               ; inače nije ništa pritisnuto, nastavi s ispitivanjem tipki

PLUS   LDR   R0[R1, #4]          ; učitati stanje tipaka sa porta B (zbog čekanja na otpuštanje tipke +)
       ANDS  R3, R0, #%B 1       ; eliminirati sve ostale bitove osim tipke +
       BNE   PLUS                ; pritisnut je +, čekaj da se otpusti

       LDR   R0, ZELJENA         ; povećaj željenu temperaturu
       ADD   R0, R0, #1
       CMP   R0, #%D 30          ; provjeri je li temperatura veća ili manja od granice 30
       STRLS R0, ZELJENA         ; ako je manja ili jednaka (LS), zapiši je u memoriju (inače nema promjene)

       B     TIPKE               ; povratak na čekanje pritiska bilo koje tipke

MINUS  LDR   R0[R1, #4]          ; učitati stanje tipaka sa porta B (zbog čekanja na otpuštanje tipke -)
       ANDS  R3, R0, #%B 10      ; eliminirati sve ostale bitove osim tipke -
       BNE   MINUS               ; pritisnut je -, čekaj da se otpusti

       LDR   R0, ZELJENA         ; smanji željenu temperaturu
       SUB   R0, R0, #1
       CMP   R0, #%D 10          ; provjeri je li temperatura veća ili manja od granice 10
       STRHS R0, ZELJENA         ; ako je veća ili jednaka (HS), zapiši je u memoriju (inače nema promjene)

       B     TIPKE               ; povratak na čekanje pritiska bilo koje tipke


;*****************************************************************

GPIO_ADR     DW    0FFFF0000     ; proizvoljne adrese vanjskih jedinica
CT_ADR       DW    0FFFF1000

TISUCU       DW    %D 1000       ; konstanta za RTC
ZELJENA      DW    %D 20         ; varijabla koja čuva željenu temperaturu
```