

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 150 minuta.

1.a (1 bod): 4-bitna ALU oduzima binarne brojeve 1011-0110. Napišite stanja zastavica poslije oduzimanja. Prijenos=____, Posudba=____, Preljev=____, Ništica=____, Predznak=____. **Mora se vidjeti način izračunavanja zastavica.**

1.b (1 bod): 5-bitni podatak 10001_2 predstavlja broj _____ u NBC-formatu, odnosno broj _____ u formatu 2^k , odnosno broj _____ u formatu s bitom za predznak. Podatak 01001_2 predstavlja broj _____ u NBC-u, odnosno broj _____ u formatu 2^k . **Mora se vidjeti postupak računanja.**

1.c (1 bod): FRISC-PIO je na adresi FFFF0000. Pretpostavite da su na bitovima 0-5 spojeni senzori temperature koji stanjem 0 signaliziraju prekoračenje dozvoljene temperature. Treba izazvati prekid ako na bilo kojem senzoru dođe do prekoračenja. PIO treba programirati da radi u načinu _____ i to tako da se prvo na adresu _____ treba poslati binarni broj _____, a zatim se na adresu _____ treba poslati binarni broj _____.

1.d (1,5 boda): Sklop FRISC-DMA treba prenijeti A000 podataka iz bezuvjetne VJ na adresi FFFF0000 u memoriju od adrese 1A00 pomoću krađe ciklusa, a kraj prijenosa javlja se prekidom. Kontrolna riječ koju treba poslati DMA-sklopu glasi: _____ (binarno). Na početnu adresu (PA) sklopa DMA treba poslati podatak _____, na adresu $PA+4_{16}$ _____, te na adresu $PA+8_{16}$ treba poslati _____. Nakon svega toga na adresu _____ od DMA-sklopa treba poslati podatak _____ da bi se pokrenuo DMA-prijenos.

1.e (0,5 boda): Ako FRISC_DMA radi **krađom ciklusa**, tada za detekciju kraja DMA-prijenosa vrijedi tvrdnja:

- a) detekcija nije potrebna za krađu ciklusa (nego samo za zaustavljanje procesora)
- b) kraj se može detektirati samo pomoću prekida
- c) kraj se može detektirati samo ispitivanjem spremnosti DMA-sklopa, jer on nema dojavu kraja posluživanja prekida
- d) kraj se može detektirati i prekidom i ispitivanjem spremnosti

2 (2,5 boda): Memorijske i ulazno-izlazne sabirnice imaju različite karakteristike. Uz svaku karakteristiku upišite "MEM" ili "IO", ovisno za koju sabirnicu vrijedi navedena tvrdnja:

- a) ova sabirnica je brza _____
- b) ova sabirnica ima malu duljinu _____
- c) ova sabirnica je prilagođena različitim brzinama rada _____
- d) ova sabirnica se lakše implementira kao asinkrona _____
- e) ova sabirnica je ARM-ova sabirnica APB _____

3.a (2 boda): ARM7 izvodi sljedeći programski odsječak. Napišite njegovo trajanje u ciklusima: _____. Pokraj svake naredbe treba napisati koliko je njeno trajanje u pojedinom izvođenju.

	MOV R0, #3	_____
	MOV R1, #0	_____
LOOP	LDRH R2, [R10], #2	_____
	ADDS R1, R1, R2	_____
	ADDHI R7, R7, #1	_____
	SUBS R0, R0, #1	_____
	BNE LOOP	_____
	STMFD R13!, {R1, R7}	_____

3.b (1 bod): Ako smo na ARM-u spremili registre naredbom `STMIA R10, {R1, R2, R3}`, tada možemo obnoviti njihove vrijednosti naredbom _____. Ako smo ih pak spremili naredbom `STMIA R10!, {R1, R2, R3}`, tada njihove vrijednosti možemo obnoviti naredbom _____.

3.c (1,5 bod): Prilikom prihvata brzog prekida FIQ, ARM automatski pohranjuje registre _____. ARM pohranu obavlja _____ (na koje mjesto). Prilikom prihvata INT3, FRISC automatski pohranjuje registre _____. FRISC pohranu obavlja _____ (na koje mjesto).

3.d (1 bod): Neki procesor ima 16-bitnu adresnu sabirnicu i 8-bitnu podatkovnu sabirnicu. Procesor ima priručnu memoriju s 4-strukom asocijativnošću i 8 skupova blokova. Svaki blok unutar skupa čuva po 256 bajtova. Neka procesor pristupa **podatku** na adresi 00100101 10010111. U kojem **skupu** se može nalaziti traženi podatak? Odgovor: _____ (napišite "adresu" skupa, u binarnoj bazi). **Zaokružite** te bitove u gornjoj adresi podatka.

4. FRISC (7 bodova) Računalo s procesorom FRISC ima CT, PIO i bezuvjetnu jedinicu BVJ (adrese im odaberite sami). CT je spojen na INT0, a na priključak CNT spojen je signal frekvencije 1 Hz. PIO je spojen na INT2.

Sa nižih 5 bitova od BVJ može se bezuvjetno pročitati NBC podatak. Ovaj podatak zadaje **koliko sekundi** treba proteći do sljedećeg prekida od sklopa CT. Svaki puta kad protekne zadani broj sekundi, u prekidnom potprogramu treba očitati novo kašnjenje sa BVJ i preprogramirati CT (prvi prekid CT treba postaviti nakon 5 sekundi). Također treba u lokaciji UKUPNO osvježavati **broj do tada primljenih prekida sa CT-a**.

Kad PIO izazove prekid, treba mu poslati sadržaj lokacije UKUPNO. PIO i CT se međusobno ne mogu prekidati, a CT ima veći prioritet.

Glavni program stalno **povećava** memorijsku lokaciju BROJAC, a treba se **zaustaviti** nakon što lokacija UKUPNO poprimi vrijednost 1000_{16} .

5. ARM (7 bodova) Za ARM napišite program koji obrađuje blok 16-bitnih podataka. Blok se nalazi u memoriji na adresi 1000_{16} i zaključen je podatkom $FOFO_{16}$.

Prvo treba za svaki podatak u bloku pozvati potprogram POTP i **zamijeniti** podatak s rezultatom koji potprogram POTP vrati. Potprogram POTP prima 16-bitni parametar pomoću **lokacije iza naredbe BL**, a vraća rezultat u nižih 16 bitova **registra R0**. POTP treba prebrajati jedinice u svom 16-bitnom parametru.

Zatim, nakon zamjene podataka, svaki par novih uzastopnih 16-bitnih podataka (nazovimo ih X i Y) treba zamijeniti s 32-bitnim podatkom koji se dobiva tako da se pomnože X i Y (kao NBC brojevi). Zbog jednostavnosti pretpostavite da je broj podataka u početnom bloku sigurno paran.

6. ARM (4 bodova) Na GPIO je spojen LCD-prikaznik kao na predavanjima. Podsjetnik: **bitovi 0-6** su izlazni - ASCII-kod znaka za prikaz (ili jedan od specijalnih znakova: 0A prikazuje znakove, 0D briše interni registar); **bit 7** je izlazni - "pozitivan impuls" označava da je znak postavljen na bitove 0-6. Napišite sljedeća dva potprograma (**glavni program ne treba pisati, kao niti inicijalizaciju GPIO-a**):

Napišite **potprogram ZNAK** koji šalje jedan ASCII-znak na vrata sklopa GPIO. Prvi parametar je ASCII-znak koji se prenosi **registrom R0**. Drugi parametar prenosi se **registrom R2**, a predstavlja **adresu od registra podataka** od GPIO-vih vrata na koja je spojen LCD. Pretpostavite da je smjer vrata prethodno već bio ispravno inicijaliziran.

Napišite **potprogram STRING** koji na LCD-u prikazuje niz znakova (string). Potprogram prima adresu niza znakova **preko stoga**, a preko **registra R2** prima adresu vrata. Za ispis pojedinog znaka treba koristiti potprogram ZNAK. Niz je zaključen NUL-znakom i sigurno je kraći od 8 znakova (tako da cijeli stane na LCD).

7. ARM (7 bodova) Računalni sustav s ARM-om ima sklop RTC i dva sklopa GPIO. Adrese sklopova trebaju biti spremjene na labelama RTC, GPIO_1 i GPIO_2 i iznose redom FFFF0000, FFFF1000 i FFFF2000.

- **RTC** je spojen na IRQ, na ulaz RTC-a spojen je signal od 1 kHz.
- Na vrata **A** sklopa **GPIO_1** spojen je temperaturni sklop (kao na predavanjima). Podsjetnik: **bitovi 0-5** su ulazni - iznos temperature u rasponu od 0 do 63_{10} ; **bit 6** je ulazni - dojava da je temperatura postavljena na bitove 0-5; **bit 7** je izlazni - "pozitivan impuls" označava da je temperatura pročitana.
- Na vrata **B** sklopa **GPIO_1** spojena je tipka na **bit 0**. Kad je pritisnuta, daje stanje 1, a inače je u stanju 0.
- Na vrata **A** sklopa **GPIO_2** spojen je LCD-prikaznik (kao na predavanjima i u prethodnom zadatku).

Svake minute treba očitati temperaturu (vrijeme mjeriti RTC-om). Ako je temperatura strogo manja od 32_{10} , na LCD-u treba prikazati tekst sa labele HLADNO, a u suprotnom tekst sa labele TOPLO. Ova dva teksta su nizovi ASCII-znakova, zaključeni su znakom NUL i kraći su od 8 znakova (pretpostavite da već postoje negdje u memoriji). Za ispis teksta koristite **potprogram STRING iz prethodnog zadatka** (ne treba ga ponovno pisati).

Glavni program treba **stalno ispitivati tipku** na GPIO_1. Ako se pritisne tipka, tada treba zabraniti RTC-u da generira daljnje prekide i treba zaustaviti glavni program.

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 šalbahtere (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 150 minuta.

1.a (1 bod): 4-bitna ALU oduzima binarne brojeve 1011-0110. Napišite stanja zastavica poslije oduzimanja. Prijenos= 1, Posudba= 0, Preljev= 1, Ništica= 0, Predznak= 0. **Mora se vidjeti način izračunavanja zastavica.**

1.b (1 bod): 5-bitni podatak 10001₂ predstavlja broj 17 u NBC-formatu, odnosno broj -15 u formatu 2'k, odnosno broj -1 u formatu s bitom za predznak. Podatak 01001₂ predstavlja broj 9 u NBC-u, odnosno broj 9 u formatu 2'k. **Mora se vidjeti postupak računanja.**

1.c (1 bod): FRISC-PIO je na adresi FFFF0000. Pretpostavite da su na bitovima 0-5 spojeni senzori temperature koji stanjem 0 signaliziraju prekoračenje dozvoljene temperature. Treba izazvati prekid ako na bilo kojem senzoru dođe do prekoračenja. PIO treba programirati da radi u načinu ispitivanja bitova i to tako da se prvo na adresu FFFF 0000 treba poslati binarni broj 00 1111, a zatim se na adresu FFFF 0000 treba poslati binarni broj 0011 1111.

1.d (1,5 boda): Sklop FRISC-DMA treba prenijeti A000 podataka iz bezuvjetne VJ na adresi FFFF0000 u memoriju od adrese 1A00 pomoću krađe ciklusa, a kraj prijenosa javlja se prekidom. Kontrolna riječ koju treba poslati DMA-sklopu glasi: 0111 (binarno). Na početnu adresu (PA) sklopa DMA treba poslati podatak FFFF0000, na adresu PA+4₁₆ 1A00, te na adresu PA+8₁₆ treba poslati A000. Nakon svega toga na adresu BA+10₁₆ ili BA+16₁₀ od DMA-sklopa treba poslati podatak bilo koji podatak da bi se pokrenuo DMA-prijenos.

1.e (0,5 boda): Ako FRISC_DMA radi **krađom ciklusa**, tada za detekciju kraja DMA-prijenosa vrijedi tvrdnja:

- a) detekcija nije potrebna za krađu ciklusa (nego samo za zaustavljanje procesora)
- b) kraj se može detektirati samo pomoću prekida
- c) kraj se može detektirati samo ispitivanjem spremnosti DMA-sklopa, jer on nema dojavu kraja posluživanja prekida
- d) kraj se može detektirati i prekidom i ispitivanjem spremnosti**

2 (2,5 boda): Memorijske i ulazno-izlazne sabirnice imaju različite karakteristike. Uz svaku karakteristiku upišite "MEM" ili "IO", ovisno za koju sabirnicu vrijedi navedena tvrdnja:

- a) ova sabirnica je brza MEM
- b) ova sabirnica ima malu duljinu MEM
- c) ova sabirnica je prilagođena različitim brzinama rada IO
- d) ova sabirnica se lakše implementira kao asinkrona IO
- e) ova sabirnica je ARM-ova sabirnica APB IO

3.a (2 boda): ARM7 izvodi sljedeći programski odsječak. Napišite njegovo trajanje u ciklusima: 32. Pokraj svake naredbe treba napisati koliko je njeno trajanje u pojedinom izvođenju.

	MOV R0, #3	<u>2+1</u>
	MOV R1, #0	<u>1</u>
LOOP	LDRH R2, [R10], #2	<u>3 3 3</u>
	ADDS R1, R1, R2	<u>1 1 1</u>
	ADDHI R7, R7, #1	<u>1 1 1</u>
	SUBS R0, R0, #1	<u>1 1 1</u>
	BNE LOOP	<u>3 3 1</u>
	STMFID R13!, {R1, R7}	<u>3</u>

3.b (1 bod): Ako smo na ARM-u spremili registre naredbom STMIA R10, {R1, R2, R3}, tada možemo obnoviti njihove vrijednosti naredbom LDMIA R10, {R1, R2, R3} ili samo LDMIA. Ako smo ih pak spremili naredbom STMIA R10!, {R1, R2, R3}, tada njihove vrijednosti možemo obnoviti naredbom LDMDB R10!, {R1, R2, R3} ili LDMDB.

3.c (1,5 bod): Prilikom prihvata brzog prekida FIQ, ARM automatski pohranjuje registre PC (ili R15) i CPSR. ARM pohranu obavlja PC u LR_fiq (ili R14_fiq) i CPSR u SPSR_fiq (na koje mjesto). Prilikom prihvata INT3, FRISC automatski pohranjuje registre PC. FRISC pohranu obavlja na stog (na koje mjesto).

3.d (1 bod): Neki procesor ima 16-bitnu adresnu sabirnicu i 8-bitnu podatkovnu sabirnicu. Procesor ima priručnu memoriju s 4-strukom asocijativnošću i 8 skupova blokova. Svaki blok unutar skupa čuva po 256 bajtova. Neka procesor pristupa **podatku** na adresi 00100101 10010111. U kojem **skupu** se može nalaziti traženi podatak? Odgovor: 101 (napišite "adresu" skupa, u binarnoj bazi). **Zaokružite** te bitove u gornjoj adresi podatka.

4. FRISC (7 bodova) Računalo s procesorom FRISC ima CT, PIO i bezuvjetnu jedinicu BVJ (adrese im odaberite sami). CT je spojen na INT0, a na priključak CNT spojen je signal frekvencije 1 Hz. PIO je spojen na INT2.

Sa nižih 5 bitova od BVJ može se bezuvjetno pročitati NBC podatak. Ovaj podatak zadaje **koliko sekundi** treba proteći do sljedećeg prekida od sklopa CT. Svaki puta kad protekne zadani broj sekundi, u prekidnom potprogramu treba očitati novo kašnjenje sa BVJ i preprogramirati CT (prvi prekid CT treba postaviti nakon 5 sekundi). Također treba u lokaciji UKUPNO osvježavati **broj do tada primljenih prekida sa CT-a**.

Kad PIO izazove prekid, treba mu poslati sadržaj lokacije UKUPNO. PIO i CT se međusobno ne mogu prekidati, a CT ima veći prioritet.

Glavni program stalno **povećava** memorijsku lokaciju BROJAC, a treba se **zaustaviti** nakon što lokacija UKUPNO poprimi vrijednost 1000_{16} .

```
CTLR    `EQU    0FFFF0000
CTCR    `EQU    0FFFF0004
CTIACK  `EQU    0FFFF0008
CTIEND  `EQU    0FFFF000C

PIO      `EQU    0FFFF1000
PIODATA  `EQU    0FFFF1004
PIOIACK  `EQU    0FFFF1008
PIOIEND  `EQU    0FFFF100C

BVJ      `EQU    0FFFF2000

        `ORG    0
        JP      GLAVNI      ; preskok preko prekidnog vektora u glavni program

        `ORG    8           ; prekidni vektor
        DW      200

GLAVNI  MOVE     10000, SP    ; inicijalizacija stoga

        ; inicijalizacija CT-a

        MOVE     %D 5, R0    ; vremenska konstanta za prvi prekid (nakon 5 sek)
        STORE    R0, (CTLR)

        MOVE     %B 11, R0    ; CT radi i daje prekide
        STORE    R0, (CTCR)

        ; inicijalizacija PIO-a

        MOVE     %B 010, R0   ; izlazni način, s prekidom
        STORE    R0, (PIO)

        ; dozvoli prekide INT0 i INT2
        MOVE     %B 11010000, SR

LOOP    LOAD     R0, (BROJAC)  ; glavna petlja
        ADD      R0, 1, R0
        STORE    R0, (BROJAC) ; povećanje lokacije BROJAC

        LOAD     R0, (UKUPNO) ; provjera lokacije UKUPNO za...
        CMP      R0, 1000     ; ...zaustavljanje glavnog programa
        JP_NE    LOOP

        HALT

; prekidni potprogram

        `ORG     200           ; adresa prekidnog potprograma
```

```

PREKIDNI  PUSH    R0                ; spremi kontekst
          PUSH    R1
          MOVE    SR, R0
          PUSH    R0

          LOAD    R0, (CTCR)        ; prvo ispitaj prioritetniji CT
          CMP     R0, 1
          JP_EQ   POSLUZI_CT        ; ako je postavio prekid, posluži ga

POSLUZI_PIO    ; inače je PIO postavio zahtjev za prekid

          STORE   R0, (PIOIACK)     ; dojava prihvata prekida

          LOAD    R0, (UKUPNO)      ; šalji PIO-u vrijednost lokacije UKUPNO
          STORE   R0, (PIODATA)

          STORE   R0, (PIOIEND)     ; dojava kraja prekida

VAN         ; izlazak iz prekidnog potprograma s obnovom konteksta
          POP     R0
          MOVE    R0, SR
          POP     R1
          POP     R0

          RETI

POSLUZI_CT    ; posluživanje CT-a

          STORE   R0, (CTIACK)     ; dojava prihvata prekida

          LOAD    R0, (BVJ)        ; pročitaj novo kašnjenje u sekundama sa BVJ
          STORE   R0, (CTLR)       ; preprogramiraj trajanje kašnjenja na CT-u

          LOAD    R0, (UKUPNO)     ; povečaj brojač prekida sa CT-a na...
          ADD     R0, 1, R0        ; ...lokaciji UKUPNO
          STORE   R0, (UKUPNO)

          STORE   R0, (CTIEND)     ; dojava kraja prekida

          JP      VAN              ; izlazak iz prekidnog potprograma

BROJAC       DW    0                ; lokacije za brojače
UKUPNO       DW    0

```

5. ARM (7 bodova) Za ARM napišite program koji obrađuje blok 16-bitnih podataka. Blok se nalazi u memoriji na adresi 1000_{16} i zaključen je podatkom $FOFO_{16}$.

Prvo treba za svaki podatak u bloku pozvati potprogram POTP i **zamijeniti** podatak s rezultatom koji potprogram POTP vrati. Potprogram POTP prima 16-bitni parametar pomoću **lokacije iza naredbe BL**, a vraća rezultat u nižih 16 bitova **registra R0**. POTP treba prebrajati jedinice u svom 16-bitnom parametru.

Zatim, nakon zamjene podataka, svaki par novih uzastopnih 16-bitnih podataka (nazovimo ih X i Y) treba zamijeniti s 32-bitnim podatkom koji se dobiva tako da se pomnože X i Y (kao NBC brojevi). Zbog jednostavnosti pretpostavite da je broj podataka u početnom bloku sigurno paran.

```

        `ORG      0

GLAVNI  MOV       R13, #10<12      ; inicijalizacija stoga
        MOV       R10, #10<8      ; adresa bloka 1000 u R10
        LDR       R1, TERM        ; terminirajući znak FOFO u R1

```

```

LOOP1    LDRH    R2, [R10]        ; čitaj podatak
          CMP     R2, R1          ; je li to znak FOFO (tj. kraj)
          BEQ     VAN1

          STR     R2, PARAM       ; znak stavi na lokaciju za parametar (ili STRH)
          BL      POTP            ;
PARAM    DW      0               ; mjesto za parametar potprograma POTP (ili DH)
          STRH    R0, [R10], #2   ; spremi rezultat preko početnog znaka, pomakni R10

          B       LOOP1          ; ponavlja za sve podatke u bloku

VAN1     MOV     R10, #10<8      ; obnovi adresu bloka za drugu petlju

LOOP2    LDRH    R2, [R10]        ; učitaj prvi broj u paru, R10 se ne mijenja
          CMP     R2, R1          ; je li to znak FOFO (tj. kraj)
          BEQ     VAN2

          LDRH    R3, [R10, #2]   ; učitaj drugi broj u paru, R10 se ne mijenja

          MUL     R4, R2, R3      ; množenje dva broja (NBC, LDRH radi proširenje...
                                ; ništicama, rezultat je sigurno unutar 32-bita)
          STR     R4, [R10], #4   ; spremu 32-bitni umnožak i pomakni R10

          B       LOOP2          ; ponavlja za sve podatke u bloku

VAN2     HALT

TERM     DW      0F0F0          ; znak za terminiranje bloka (ne može se...
                                ; ...napisati u naredbi za obradu podataka)

```

; potprogram: POTP, prebraja jedinice u 16-bitnom podatku
 ; parametri: jedan parametar koji se nalazi iza naredbe BL, podatak za obradu
 ; rezultat: broj jedinica, vraća se registrom R0

```

POTP     STMFD   R13!, {R1,R2}   ; spremi kontekst

          LDR     R1, [LR], #4    ; dohvati parametar i pomakni povratnu adresu...
                                ; ...na mjesto za povratak (tj. na naredbu iza...
                                ; ...parametra)

          MOV     R2, #D 16       ; brojač za petlju
          MOV     R0, #0          ; brojač jedinica, ujedno i rezultat potprograma
          MOVS    R1, R1, ROR #1  ; najniži bit pomaknu u zastavicu C
          ADDCS   R0, R0, #1      ; bit==1 => povećaj brojač jedinica
                                ; (može i ADC R0,R0,#0)
          SUBS    R2, R2, #1      ; ponavlja petlju 16 puta
          BNE     LOOP

          LDMFD   R13!, {R1,R2}
          MOV     PC, LR

```

6. ARM (4 bodova) Na GPIO je spojen LCD-prikaznik kao na predavanjima. Podsjetnik: **bitovi 0-6** su izlazni - ASCII-kod znaka za prikaz (ili jedan od specijalnih znakova: 0A prikazuje znakove, 0D briše interni registar); **bit 7** je izlazni - "pozitivan impuls" označava da je znak postavljen na bitove 0-6. Napišite sljedeća dva potprograma (**glavni program ne treba pisati, kao niti inicijalizaciju GPIO-a**):

Napišite **potprogram ZNAK** koji šalje jedan ASCII-znak na vrata sklopa GPIO. Prvi parametar je ASCII-znak koji se prenosi **registrom R0**. Drugi parametar prenosi se **registrom R2**, a predstavlja **adresu od registra podataka** od GPIO-vih vrata na koja je spojen LCD. Pretpostavite da je smjer vrata prethodno već bio ispravno inicijaliziran.

Napišite **potprogram STRING** koji na LCD-u prikazuje niz znakova (string). Potprogram prima adresu niza znakova **preko stoga**, a preko **registra R2** prima adresu vrata. Za ispis pojedinog znaka treba koristiti potprogram ZNAK. Niz je zaključen NUL-znakom i sigurno je kraći od 8 znakova (tako da cijeli stane na LCD).

```
;potprogram STRING: ispis niza znakova (stringa) na LCD
;parametri: R1=adresa niza, R2=adresa porta s LCD-om
```

```
STRING    STMFD    R13!, {R0, R1, LR}        ; spremanje konteksta, R2 se ne mijenja

          LDR      R1, [R13, #0C]             ; učitaj parametar (adresu niza)...
          ; ...sa stoga u registar R1

          MOV      R0, #0D
          BL       ZNAK                       ; briši LCD (u R2 je već ispravan parametar)

PETLJA    LDRB     R0, [R1], #1               ; čitaj znak po znak iz niza i pomiči R1
          CMP      R0, #0
          BEQ      GOTOVO                     ; ako je NUL-znak, ispis je gotov

          BL       ZNAK                       ; inače ispiši znak
          B        PETLJA                     ; idi na sljedeći znak

GOTOVO    MOV      R0, #0A
          BL       ZNAK                       ; prikaži poslane znakove na LCD

          LDMFD    R13!, {R0, R1, LR}        ; obnova konteksta s povratkom
          MOV      PC, LR
```

```
;potprogram ZNAK: slanje jednog znaka na LCD
;parametri: R0=znak, R2=adresa porta s LCD-om
```

```
ZNAK      STMFD    R13!, {R0}                ; spremanje konteksta

          AND      R0, R0, #%B 01111111      ; obriši bit 7 u znaku (nije nužno)
          STR      R0, [R2]                  ; pošalji ASCII na vrata

          ; generiranje impulsa na bitu 7 - upis znaka
          ORR      R0, R0, #%B 10000000      ; digni bit 7 na vratima
          STR      R0, [R2]

          AND      R0, R0, #%B 01111111      ; spusti bit 7 na vratima
          STR      R0, [R2]

          LDMFD    R13!, {R0}                ; obnova konteksta s povratkom
          MOV      PC, LR
```

7. ARM (7 bodova) Računalni sustav s ARM-om ima sklop RTC i dva sklopa GPIO. Adrese sklopova trebaju biti spremljene na labelama RTC, GPIO_1 i GPIO_2 i iznose redom FFFF0000, FFFF1000 i FFFF2000.

- **RTC** je spojen na IRQ, na ulaz RTC-a spojen je signal od 1 kHz.
- Na vrata **A** sklopa **GPIO_1** spojen je temperaturni sklop (kao na predavanjima). Podsjetnik: **bitovi 0-5** su ulazni - iznos temperature u rasponu od 0 do 63₁₀; **bit 6** je ulazni - dojava da je temperatura postavljena na bitove 0-5; **bit 7** je izlazni - "pozitivan impuls" označava da je temperatura pročitana.
- Na vrata **B** sklopa **GPIO_1** spojena je tipka na **bit 0**. Kad je pritisnuta, daje stanje 1, a inače je u stanju 0.
- Na vrata **A** sklopa **GPIO_2** spojen je LCD-prikaznik (kao na predavanjima i u prethodnom zadatku).

Svake minute treba očitati temperaturu (vrijeme mjeriti RTC-om). Ako je temperatura strogo manja od 32₁₀, na LCD-u treba prikazati tekst sa labele HLADNO, a u suprotnom tekst sa labele TOPLO. Ova dva teksta su nizovi ASCII-znakova, zaključeni su znakom NUL i kraći su od 8 znakova (pretpostavite da već postoje negdje u memoriji). Za ispis teksta koristite **potprogram STRING iz prethodnog zadatka** (ne treba ga ponovno pisati).

Glavni program treba **stalno ispitivati tipku** na GPIO_1. Ako se pritisne tipka, tada treba zabraniti RTC-u da generira daljnje prekide i treba zaustaviti glavni program.

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

`ORG      18
B PREKIDNI                                  ; prekidni potprogram - IRQ
; (skok može i ne mora jer se FIQ ne koristi)
```

```

GLAVNI    MOV        R13,#10<12                ; inicijalizacija stoga

; inicijalizacija RTC-a da generira prekide svake minuta
LDR        R0, RTC                            ; bazna adresa sklopa RTC

LDR        R1, KONST                          ; konstanta brojenja 60000
STR        R1, [R0,#4]                        ; upis u RTCMR

MOV        R1, #1                             ; omogućavanje prekida u RTC-u
STR        R1, [R0,#10]                       ; upis u upravljački registar (RTCCR)

MOV        R1, #0                             ; inicijalizacija brojila (nije nužno)
STR        R1, [R0,#0C]                       ; upis u RTCLR

; inicijalizacija PIO_1 za temperaturni sklop i tipku
LDR        R0, GPIO_1                        ; bazna adresa sklopa GPIO_1

MOV        R1, %%B10000000                    ; bit 7 je izlazni (za temp.sklop)
STR        R1, [R0,#8]                        ; registar smjera vrata A

MOV        R1, %%B 00000001                    ; bit 0 je ulazni (za tipku)
STR        R1, [R0,#0C]                       ; registar smjera vrata B

; inicijalizacija PIO_2 za LCD
LDR        R0, GPIO_2                        ; bazna adresa sklopa GPIO_2

MOV        R1, %%B11111111                    ; svi bitovi izlazni (za LCD)
STR        R1, [R0,#8]                        ; registar smjera vrata A

MRS        R0, CPSR                           ; omogućavanje prihvata IRQ-a
BIC        R0, R0, #80
MSR        CPSR_c, R0

LOOP      LDR        R0, GPIO_1                ; bazna adresa sklopa GPIO_1
          LDR        R1, [R0, #4]              ; učitaj stanje tipke
          CMP        R1, #1
          BNE        LOOP                      ; vrti petlju dok tipka nije pritisnuta

STOP      LDR        R0, RTC
          MOV        R1, #0
          STR        R1, [R0,#10]              ; zaustavi RTC

          HALT

RTC        DW 0FFFF0000
GPIO_1     DW 0FFFF1000
GPIO_2     DW 0FFFF2000
KONST      DW %D 60000

PREKIDNI   STMFD R13!, {R0,R1,R2,LR}          ; spremanje konteksta

          LDR        R0, RTC                  ; učitavanje adrese RTC-a

; reinicijalizacija RTC-a
MOV        R1, #0
STR        R1, [R0,#0C]                      ; resetiranje brojača
STR        R1, [R0,#8]                        ; dojava prihvata prekida

; čitanje temperature (na vratima A)
LDR        R0, GPIO_1                        ; učitavanje adrese GPIO_1

CEKAJ      LDR        R1, [R0, #0]              ; čekanje na novu temperaturu
          TST        R1, %%B 01000000          ; bit 6 je signal nove temp.
          BEQ        CEKAJ                    ; čekanje dok je signal u niskom stanju

          ORR        R1, R1, %%B 10000000      ; generiranje impulsa na bitu 7...

```


STR	R1, [R0]	; (umjesto OR/AND može i MOV 80 pa MOV 0...
AND	R1, R1, #%B 01111111	; ... ali s dodatnim registrom)
STR	R1, [R0]	
AND	R1, R1, #%B 00111111	; izdvoji samo bitove 0-5 s temperaturom
CMP	R1, #D 32	; odredi string za ispis u ovisnosti...
		; ...o temperaturi (veća ili manja od 32)
MOVLO	R1, #HLADNO	; adresu odgovarajućeg stringa stavi u R1...
MOVHS	R1, #TOPLO	; ...kao parametar, uz pretpostavku da te...
		; ...adrese stanu u pomaknutih 8 bita, inače
		; ...bi trebalo koristiti pseudonaredbu ADR.
STMFD	R13!, {R1}	; parametar iz R1 stavi na <u>stog</u> (1. param.)
LDR	R2, GPIO_2	; učitavanje adrese GPIO_2 (za LCD)
ADD	R2, R2, #4	; adresu porta B stavi u <u>R2</u> (2. parametar)
BL	STRING	; ispiši string
ADD	R13, R13, #4	; počisti parametar sa stoga
VAN	LDMFD	R13!, {R0,R1,R2,LR} ; obnova konteksta
	SUBS	PC, LR, #4 ; povratak