Završni ispit iz Arhitekture računala 1 — 27. lipnja 2017.

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

Izjavljujem da tijekom ispita 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 i nosi 40 bodova.

1.a (1,5 bod) 4-bitni podatak 1001_2 u formatu **NBC** predstavlja broj _____, u formatu **2'k** predstavlja broj _____, te u formatu **s bitom za predznak** predstavlja broj _____. **4-bitni podatak 0101_2** u formatu **NBC** predstavlja broj_____, u formatu **2'k** predstavlja broj _____, te u formatu s **bitom za predznak** predstavlja broj_____. **Mora se vidjeti postupak.**

Mjesto za postupak:

1.b (1 bod) Za **memorijske i ulazno-izlazne (UI)** sabirnice vrijede sljedeće tvrdnje. **Veća brzina rada** je karakteristika _____ sabirnice. Spajanje **većeg broja uređaja** različitih brzina rada je karakteristika _____ sabirnice. Dva **načina povezivanja** memorijske i ulazno-izlazne sabirnice su:_____ i _____.

1.c (1 bod) FRISC-GPIO se spaja sa **vanjskim procesom** pomoću sinkronizacijskih priključaka nazvanih _____ i _____. Ovi priključci se koriste u dva **načina rada** koja se zovu _____ i _____.

1.d (2,5 boda) Tri **načina** DMA-prijenosa su: _____, _____ i _____.
Ako se u sklopu **FRISC-DMA** zada da je **izvor** podataka **memorija**, onda će se nakon prijenosa svakog podatka automatski **za 4 uvećati** _____. To se **neće** dogoditi ako se zada da je izvor _____. Bez obzira na vrstu izvora i odredišta, nakon prijenosa svakog podatka **uvijek će se za 1 smanjiti** _____. Sklop FRISC-DMA postat će **spreman** (kada?) _____ _____.

Ako je sklop FRISC-DMA na adresi **FFFF0000** i ako želimo **kopirati blok** memorije od 100 podataka, od adrese 1000 na adresu 2000, **krađom ciklusa** i **bez prekida**, onda 4 najniža bita **kontrolne riječi** moraju biti _____ **(binarno)**, a tu riječ treba upisati na **adresu** _____. U ovom slučaju **detekciju kraja** DMA-prijenosa napravit ćemo čitanjem i ispitivanjem podatka s adrese _____.

1.e (1 bod) Za ARM napišite programski odsječak (najviše 4 naredbe) koji **uspoređuje dva 2'k broja u dvostrukoj preciznosti**. **Prvi broj** je u R0 (niži dio) i R1 (viši dio). **Drugi broj** je u R2 (niži dio) i R3 (viši dio). Ako je prvi broj **veći ili jednak** od drugoga, onda u R6 treba upisati 1, a inače u R6 treba upisati 0. Niti jedan opći registar osim R6 (i R15, naravno) **ne smije** promijeniti vrijednost.

_____
_____
_____
_____

1.f (3 boda) Za **ARM7** odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**). Kraj **svake** naredbe napišite **koliko puta** se naredba izvodi po **koliko** ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c). Ako neka naredba izaziva **hazard**, napišite njegovo ime na drugoj crti. Cijeli odsječak traje _____ ciklusa.

```
        B     SKIP
TRI     DW    3
SKIP    LDR   R6, TRI
        MOV   R0, #9
LOOP    SUBS  R0, R0, R6
        BNE   LOOP
        STR   R0, REZ
```

Trajanje	Hazard
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

1.g (2 boda) FRISC prilikom **prihvaćanja** prekida **NMI** automatski sprema _____ _____ (što?) na _____ (gdje?). Prekidni potprogram za NMI se nalazi na adresi _____.

ARM prilikom **prihvaćanja** prekida **FIQ** automatski sprema _____ (što?) u _____(gdje?), a također automatski sprema i _____ (što?) u _____ (gdje?). Prekidni potprogram za FIQ se nalazi na adresi _____.

1.h (1 bod) Efikasnost priručnih memorija (cache) temelji se na dvije **karakteristike podataka** u stvarnim primjenama. Te **karakteristike podataka** se nazivaju _____ i _____. Prilikom **pisanja bez promašaja** u priručnu memoriju treba osigurati **istovjetnost** (koherentnost) podataka. Dva algoritma za obradu pisanja zovu se: _____ i _____.

2. FRISC (10 bodova) Na FRISC su spojeni sklopovi CT1 i CT2 te GPIO1 i GPIO2. Adrese im odaberite sami.

Pomoću sklopa **CT2** treba **mjeriti razdoblja od 1 ms** (milisekunde). Na ulaz od CT2 je spojen signal frekvencije 1 MHz. Sklop CT2 treba zahtijevati prekid preko **NMI**.

CT1 treba **prebrajati impulse** koji dolaze na njegov ulaz. Prebraja se **koliko je impulsa došlo unutar prethodnog razdoblja od 1 ms**. Pretpostavite da broj impulsa unutar 1 ms neće prijeći 8-bitni opseg. Nakon što se se prebroje impulsi iz prethodne milisekunde, onda taj broj treba **bezuvjetno poslati na GPIO1**.

Brojenje impulsa treba napraviti **100_{10} puta**, tj. za 100_{10} razdoblja od jedne milisekunde. Nakon toga treba **zaustaviti rad oba CT-a**.

GPIO2 izaziva prekide **INT**. Kad se dogodi INT, na GPIO2 treba poslati **broj 2 ako CT-ovi obavljaju svoju funkciju**. Ako su pak **CT-ovi zaustavljeni**, onda na GPIO2 treba poslati **broj 0**.

Glavni program cijelo vrijeme izvodi **beskonačnu petlju**.

3. ARM (7 bodova) Treba napisati potprogram PRETVORI koji **pretvara broj iz 32-bitnog formata s bitom za predznak u 32-bitni format 2'k**. Potprogram prima jedan **parametar** preko **lokacije iza naredbe BL**. Parametar predstavlja **adresu** broja u formatu s bitom za predznak. **Rezultat** se vraća registrom **R2**.

Treba napisati potprogram SUM_BLOK koji **računa sumu svih podataka u memorijskom bloku**. SUM_BLOK prima **dva parametra preko stoga**: početnu **adresu** bloka i **broj** podataka u bloku. Podatci u bloku su u **32-bitom formatu s bitom za predznak**. Za **svaki podatak** treba **pomoću potprograma PRETVORI** odrediti prikaz u formatu 2'k i **pribrojiti ga sumi** koja također mora biti u **formatu 2'k**. **Suma mora biti u dvostrukoj preciznosti**. Rezultat potprograma je ukupna suma svih podataka, pri čemu se **niži dio sume vraća registrom R0, a viši dio registrom R1**.

Glavni program treba samo pozvati potprogram SUM_BLOK za blok na adresi 45A0 u kojemu se nalazi 3000_{16} podataka u 32-bitnom formatu s bitom za predznak. Rezultat treba spremiti od lokacije 200_{16}.

4. ARM (10 bodova) Na ARM su spojeni GPIO i RTC (adrese odaberite sami). Na **RTC** je spojen signal od **1kHz**, a RTC je spojen na **FIQ**.

Na **vrata A** sklopa GPIO spojen je **LCD prikaznik** kao na predavanjima (podsjetnik: bitovi 0 do 6=ASCII znak, bit 7=WR, izlazni). Na **vrata B** na bit 7 spojena je **sklopka** (daje 1 kada je pritisnuta), a na bitove 0 do 6 spojen je **senzor** tlaka (stanje senzora se očitava bezuvjetno).

Svake sekunde treba očitati stanje sklopke. Ako je uključena, treba na LCD-u prikazati tekst „ON“, a ako je isključena, prikaz na LCD-u treba biti prazan. ASCII kodovi: 'O'=4F, 'N'=4E, LCD briši=0D, LCD prikaži=0A.

Napišite **potprogram LCDWR** (kao na predavanjima) pomoću kojega ćete **slati pojedini znak na LCD** (način prijenosa parametara odaberite po svojoj želji, ali napišite u komentaru kako ih prenosite).

Glavni program **cijelo vrijeme treba očitavati stanje senzora** na GPIO-u, i pohranjivati ga kao 8-bitni podatak **kružno** u memorijski blok od 100_{16} **bajtova** na adresi 1000_{16}. **Kružno pohranjivanje** znači da se nakon što se blok **napuni**, sljedeći **bajt** opet puni od **početne** adrese.

RJEŠENJA

1.a (1,5 bod) **4-bitni podatak 1001_2** u formatu **NBC** predstavlja broj **9**, u formatu **2'k** predstavlja broj **-7**, te u formatu **s bitom za predznak** predstavlja broj **-1**. **4-bitni podatak 0101_2** u formatu **NBC** predstavlja broj **5**, u formatu **2'k** predstavlja broj **5**, te u formatu s **bitom za predznak** predstavlja broj **5**. **Mora se vidjeti postupak.**

Mjesto za postupak:	**NBC:** **1001 = 8+1 = 9**	**2'k:** **1001 = negativan -(0110+1)= (0111) = -(4+2+1)= -7**	**Bit za predznak:** **1001 = negativan -(0001) = -(1)=-1**	**0101 je pozitivan pa ima isti iznos za sva tri formata: 0101 = 4+1 = 5**

1.b (1 bod) Za **memorijske i ulazno-izlazne (UI)** sabirnice vrijede sljedeće tvrdnje. **Veća brzina rada** je karakteristika **memorijske** sabirnice. Spajanje **većeg broja uređaja** različitih brzina rada je karakteristika **ulazno-izlazne** sabirnice. Dva **načina povezivanja** memorijske i ulazno-izlazne sabirnice su: **zajednička sabirnica ili backplane** i **neizravno spajanje ili spajanje pomoću međusklopa/mosta**.

1.c (1 bod) **FRISC-GPIO** se spaja sa **vanjskim procesom** pomoću sinkronizacijskih priključaka nazvanih **STROBE** i **READY**. Ovi priključci se koriste u dva **načina rada** koja se zovu **ulazni (način)** i **izlazni (način)**.

1.d (2,5 boda) Tri **načina** DMA-prijenosa su: **krađa ciklusa**, **zaustavljanje procesora** i **blokovski (prijenos)**.
Ako se u sklopu **FRISC-DMA** zada da je **izvor** podataka **memorija**, onda će se nakon prijenosa svakog podatka automatski **za 4 uvećati** **adresni registar izvora/adresa izvora**. To se **neće** dogoditi ako se zada da je izvor **vanjska jedinica**. Bez obzira na vrstu izvora i odredišta, nakon prijenosa svakog podatka **uvijek će se za 1 smanjiti** **brojač ili brojač (prenesenih) podataka**. Sklop FRISC-DMA postat će **spreman** (kada?) **kad se svi podatci prenesu/kad se dovrši DMA-prijenos/kad brojač postane 0**.

Ako je sklop FRISC-DMA na adresi **FFFF0000** i ako želimo **kopirati blok** memorije od 100 podataka, od adrese 1000 na adresu 2000, **krađom ciklusa** i **bez prekida**, onda 4 najniža bita **kontrolne riječi** moraju biti **0010** **(binarno)**, a tu riječ treba upisati na **adresu** **FFFF000C**. U ovom slučaju **detekciju kraja** DMA-prijenosa napravit ćemo čitanjem i ispitivanjem podatka s adrese **FFFF0014 (adresa bistabila stanja) i/ili FFFF0008 (adresa brojača podataka)**.

1.e (1 bod) Za ARM napišite programski odsječak (najviše 4 naredbe) koji **uspoređuje dva 2'k broja u dvostrukoj preciznosti**. **Prvi broj** je u R0 (niži dio) i R1 (viši dio). **Drugi broj** je u R2 (niži dio) i R3 (viši dio). Ako je prvi broj **veći ili jednak** od drugoga, onda u R6 treba upisati 1, a inače u R6 treba upisati 0. Niti jedan opći registar osim R6 (i R15, naravno) **ne smije** promijeniti vrijednost.

```
SUBS R6,R0,R2
SBCS R6,R1,R3
MOVGE R6,#1
MOVLT R6,#0
```

Uočite da sljedeća rješenja nisu dobra, tj. ne rade za sve brojeve (probajte sa npr. 1111 1000 i 1111 0000 u 4-bitnoj jednostrukoj preciznosti i 8-bitnoj dvostrukoj preciznosti):

```
MOV  R6,#0          CMP  R1, R3
CMP  R1, R3         CMPEQ  R0, R2
CMPEQ  R0, R2       MOVGE  R0, #1
MOVGE  R6, #1       MOVLT  R0, #0
```

1.f (3 boda) Za **ARM7** odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**). Kraj **svake** naredbe napišite **koliko puta** se naredba izvodi po **koliko** ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c). Ako neka naredba izaziva **hazard**, napišite njegovo ime na drugoj crti. Cijeli odsječak traje **21** ciklusa.

```
        B     SKIP
TRI     DW    3
SKIP    LDR   R6, TRI
        MOV   R0, #9
LOOP    SUBS  R0, R0, R6
        BNE   LOOP
        STR   R0, REZ
```

	Trajanje	Hazard
	2punj + 1x3c	**upravljački**
	1x3c	**strukturni**
	1x1c	
	3x1c	
	2x3c + 1x1c	**upravljački**
	1x2c	**strukturni**

1.g (2 boda) **FRISC** prilikom **prihvaćanja** prekida **NMI** automatski sprema **(registar) PC ili povratnu adresu (=adresa sljedeće naredbe)** (što?) na **stog** (gdje?). Prekidni potprogram za NMI se nalazi na adresi **0C**.

ARM prilikom **prihvaćanja** prekida **FIQ** automatski sprema **PC ili povratnu adr.+4** (što?) u **LR_fiq/R14_fiq** (gdje?), a također automatski sprema i **CPSR** (što?) u **SPSR_fiq** (gdje?). Prekidni potprogram za FIQ se nalazi na adresi **1C**.

1.h (1 bod) Efikasnost priručnih memorija (cache) temelji se na dvije **karakteristike podataka** u stvarnim primjenama. Te **karakteristike podataka** se nazivaju **prostorna lokalnost** i **vremenska lokalnost**. Prilikom **pisanja bez promašaja** u priručnu memoriju treba osigurati **istovjetnost** (koherentnost) podataka. Dva algoritma za obradu pisanja zovu se: **pisanje s prosljeđivanjem/write-through** i **pisanje pri povratku/write-back**.

2. FRISC (10 bodova) Na FRISC su spojeni sklopovi CT1 i CT2 te GPIO1 i GPIO2. Adrese im odaberite sami.

Pomoću sklopa **CT2** treba **mjeriti razdoblja od 1 ms** (milisekunde). Na ulaz od CT2 je spojen signal frekvencije 1 MHz. Sklop CT2 treba zahtijevati prekid preko **NMI**.

CT1 treba **prebrajati impulse** koji dolaze na njegov ulaz. Prebraja se **koliko je impulsa došlo unutar prethodnog razdoblja od 1 ms**. Pretpostavite da broj impulsa unutar 1 ms neće prijeći 8-bitni opseg. Nakon što se se prebroje impulsi iz prethodne milisekunde, onda taj broj treba **bezuvjetno poslati na GPIO1**.

Brojenje impulsa treba napraviti **100_{10} puta**, tj. za 100_{10} razdoblja od jedne milisekunde. Nakon toga treba **zaustaviti rad oba CT-a**.

GPIO2 izaziva prekide **INT**. Kad se dogodi INT, na GPIO2 treba poslati **broj 2 ako CT-ovi obavljaju svoju funkciju**. Ako su pak **CT-ovi zaustavljeni**, onda na GPIO2 treba poslati **broj 0**.

Glavni program cijelo vrijeme izvodi **beskonačnu petlju**.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; adrese, početak i vektor


CT1_CR     EQU   0FFFF1000            ; adrese vanjskih jedinica
CT1_LR     EQU   0FFFF1004

CT2_CR     EQU   0FFFF2000
CT2_LR     EQU   0FFFF2004
CT2_IACK   EQU   0FFFF2008
CT2_IEND   EQU   0FFFF200C

GPIO1_CR   EQU   0FFFF3000
GPIO1_DATA EQU   0FFFF3004

GPIO2_CR   EQU   0FFFF4000
GPIO2_DATA EQU   0FFFF4004
GPIO2_IACK EQU   0FFFF4008
GPIO2_IEND EQU   0FFFF400C

           ORG   0
           MOVE  10000, SP           ; inicijalizacija stoga
           JP    GLAVNI              ; preskok u glavni

           ORG   8                   ; adresa vektora
           DW    200                 ; vektor


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; prekidni potprogram za NMI
; CT2 JAVLJA DA JE ISTEKLA 1 ms

           ORG   0C                  ; adresa za NMI
NMI_PP     PUSH  R0                  ; pohrani kontekst
           PUSH  R1
           MOVE  SR, R0
           PUSH  R0

           STORE R0, (CT2_IACK)      ; prihvat prekida

           ; učitaj i resetiraj iznos brojila u CT1
           LOAD  R1, (CT1_LR)        ; učitaj iznos brojila i zapamti ga u R1
           MOVE  0, R0                 ; resetiraj brojilo:
           STORE R0, (CT1_LR)          ; koristim konstantu 0, kao i u glavnom programu

           ; izračunaj broj impulsa u prošloj milisekundi. Koristi se konstanta
           ; 65536 jer je inicijalno u LR upisan broj 0. Može i neka druga konstanta.
           MOVE  %D 65536, R0
           SUB   R0, R1, R0
```

```
          ; pošalji izračunati broj impulsa na GPIO1 (i to bezuvjetno)
          STORE R0, (GPIO1_DAT)

          LOAD  R0, (BROJAC_100)       ; smanji brojač milisekundi
          SUB   R0, 1, R0
          STORE R0, (BROJAC_100)

          CMP   R0, 0                 ; ispitaj brojač milisekundi (je li 100)
          JR_NE NASTAVI

          ; proteklo je 100 milisekundi - zaustavi CT-ove i postavi oznaku
KRAJ      MOVE  0, R0
          STORE R0, (CT1_CR)
          STORE R0, (CT2_CR)
          STORE R0, (OZNAKA)

NASTAVI   STORE R0, (CT2_IEND)       ; dojava kraja posluživanja

          POP   R0                    ; obnovi kontekst
          MOVE  R0, SR
          POP   R1
          POP   R0
          RETN                        ; povratak iz NMI



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; glavni program

GLAVNI    MOVE %D 1000, R0             ; inicijalizacija CT2 (1 ms)
          STORE R0, (CT2_LR)
          MOVE %B 111, R0              ; NMI, prekid = da; START
          STORE R0, (CT2_CR)

          MOVE 0, R0                   ; inicijalizacija CT1 (counter)
          STORE R0, (CT1_LR)
          ; Ne mora se upisati 0 u LR, već bilo koja konstanta veća od 255, ali
          ; onda tu konstantu treba koristiti kod izračuna broja impulsa u
          ; prekidnom potprogramu za NMI.
          MOVE %B 01, R0               ; prekid = ne; START
          STORE R0, (CT1_CR)

          ; inicijalizacija GPIO1
          MOVE %B 010, R0              ; prekid = ne; MOD = postavljanje bitova
          STORE R0, (GPIO1_CR)

          ; inicijalizacija GPIO2
          MOVE %B 0100, R0             ; INT, prekid = da; MOD = izlazni
          STORE R0, (GPIO2_CR)

          MOVE  %B 10000, SR           ; dozvoli prekide

LOOP      JR    LOOP                  ; beskonačna petlja



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; dvije varijable
; obavezno u memoriji, ne držati po registrima

BROJAC_100 DW    %D 100               ; brojač milisekundi
OZNAKA     DW    2                    ; stanje rada CT-ova: rade oba (2), ne rade (0)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; prekidni potprogram za INT

INT_PP     ORG   200                  ; adresa prekidnog potprograma

           PUSH  R0                   ; pohrana konteksta (SR se ne mijenja)

           STORE R0, (GPIO2_IACK)     ; prihvat prekida

           LOAD  R0, (OZNAKA)         ; dohvat OZNAKE (stanje rada CT_ova)
           STORE R0, (GPIO2_DATA)     ; slanje OZNAKE (broj 0 ili 2) na GPIO2

           STORE R0, (GPIO2_IEND)     ; dojava kraja posluživanja

           POP   R0                   ; obnova konteksta
           RETI                       ; povratak iz INT
```

3. ARM (7 bodova) Treba napisati potprogram PRETVORI koji **pretvara broj iz 32-bitnog formata s bitom za predznak u 32-bitni format 2'k**. Potprogram prima jedan **parametar** preko **lokacije iza naredbe BL**. Parametar predstavlja **adresu** broja u formatu s bitom za predznak. **Rezultat** se vraća registrom **R2**.

Treba napisati potprogram SUM_BLOK koji **računa sumu svih podataka u memorijskom bloku**. SUM_BLOK prima **dva parametra preko stoga**: početnu **adresu** bloka i **broj** podataka u bloku. Podatci u bloku su u **32-bitom formatu s bitom za predznak**. Za **svaki podatak** treba **pomoću potprograma PRETVORI** odrediti prikaz u formatu 2'k i **pribrojiti ga sumi** koja također mora biti u **formatu 2'k**. **Suma mora biti u dvostrukoj preciznosti**. Rezultat potprograma je ukupna suma svih podataka, pri čemu se **niži dio sume vraća registrom R0, a viši dio registrom R1**.

Glavni program treba samo pozvati potprogram SUM_BLOK za blok na adresi 45A0 u kojemu se nalazi 3000_{16} podataka u 32-bitnom formatu s bitom za predznak. Rezultat treba spremiti od lokacije 200_{16}.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; glavni program i podatci

            ORG   0
GLAVNI      MOV   SP, #10<12              ; inicijalizacija stoga

            LDR   R0, ADR_NIZA            ; adresa 45A0 ne može naredbom MOV
            MOV   R1, #30<8               ; N=3000 podataka u bloku
            STMFD SP!, {R0,R1}            ; parametri na stog

            BL    SUM_BLOK
            ADD   SP, SP, #8              ; ukloni parametre sa stoga

            MOV   R2, #20<4               ; spremanje rezultata iz R0 i R1 na adresu 200
            STR   R0, [R2], #4
            STR   R1, [R2]
            ; umjesto dve gornje naredbe može i STMIA R2, {R0, R1}

            ; budući da je 200 blizu za 12-bitni ofset, onda umjesto gornje tri naredbe
            ; može i:   STR R0,200; STR R1,204


            SWI   123456



ADR_NIZA    DW 45A0

            ORG 200
            DW 0, 0      ; mjesto za rezultat
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; potprogram SUM_BLOK

SUM_BLOK   STMFD SP!, {R2, R3, R4, LR}          ; spremanje konteksta
                                                ; R2 se sprema jer ga PRETVORI mijenja
                                                ; LR se sprema zbog BL

           ADD   R2, SP, #%D 16           ; R2=adresa parametara
           LDMFD R2, {R3, R4}             ; dohvat parametara
                                          ;R3=adresa bloka, R4=broj podataka

           MOV   R0, #0                   ; rezultat, suma u dvostrukoj preciznosti
           MOV   R1, #0

LOOP       STR   R3, ADR                  ; upiši parametar iza naredbe BL
           BL    PRETVORI                 ; poziv
ADR        DW    0                        ; mjesto za  parametar

        ; Pribrajanje broja u sumu (koja je u R0 i R1 u dvostrukoj preciznosti).
        ; Broj je u R2 kao 32-bitni 2'k, treba ga proširiti pri pribrajanju višeg dijela.
           ADDS  R0, R0, R2               ; pribroji niži dio broja sumi
           MOV   R2, R2, ASR #%D 31       ; proširi, tj. odredi viši dio broja u kojemu
                                          ; su sve 0 ili sve 1
           ADC   R1, R1, R2               ; pribroji viši dio sumi

           ADD   R3, R3, #4               ; izračunaj adresu sljedećeg podatka
           SUBS  R4, R4, #1               ; smanji brojač petlje, tj. broj podataka
           BNE   LOOP

           LDMFD SP!, {R2, R3, R4, LR}       ; obnovi kontekst
           MOV   PC, LR                      ; povratak
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; potprogram PRETVORI

PRETVORI   STMFD SP!, {R0}           ; spremi kontekst

           LDR   R0, [LR], #4             ; dohvat parametra i pomak povratne adrese
                                          ;R0=adresa podatka
           LDR   R2, [R0]      ; dohvati broj sa zadane adrese

           ORRS  R2, R2, R2    ; ispitaj predznak broja

           BPL   POZIT_VAN           ; ako je pozitivan, idi na POZIT_VAN
NEGAT      BIC   R2, R2, #80<24      ; obriši bit predznaka
           RSB   R2, R2, #0          ; operacija dvojnog komplementa

           ; Gornje tri naredbe mogu i kraće: BICMI i RSBMI
           ; umjesto RSB može i klasično "EOR -1" i "ADD 1"

POZIT_VAN  LDMFD SP!, {R0}           ; obnovi kontekst
           MOV   PC, LR              ; povratak

; Može i bez spremanja konteksta, jer se sve može riješiti samo s
; registrom R2. Prve dvije naredbe mogu se napisati ovako:
;          LDR   R2, [LR], #4       ; R2=adresa podatka
;          LDR   R2, [R2]     ; dohvati broj sa zadane adrese
```

4. ARM (10 bodova) Na ARM su spojeni GPIO i RTC (adrese odaberite sami). Na **RTC** je spojen signal od **1kHz**, a RTC je spojen na **FIQ**.

Na **vrata A** sklopa GPIO spojen je **LCD prikaznik** kao na predavanjima (podsjetnik: bitovi 0 do 6=ASCII znak, bit 7=WR, izlazni). Na **vrata B** na bit 7 spojena je **sklopka** (daje 1 kada je pritisnuta), a na bitove 0 do 6 spojen je **senzor** tlaka (stanje senzora se očitava bezuvjetno).

Svake sekunde treba očitati stanje sklopke. Ako je uključena, treba na LCD-u prikazati tekst „ON“, a ako je isključena, prikaz na LCD-u treba biti prazan. ASCII kodovi: 'O'=4F, 'N'=4E, LCD briši=0D, LCD prikaži=0A.

Napišite **potprogram LCDWR** (kao na predavanjima) pomoću kojega ćete **slati pojedini znak na LCD** (način prijenosa parametara odaberite po svojoj želji, ali napišite u komentaru kako ih prenosite).

Glavni program **cijelo vrijeme treba očitavati stanje senzora** na GPIO-u, i pohranjivati ga kao 8-bitni podatak **kružno** u memorijski blok od 100_{16} **bajtova** na adresi 1000_{16}. **Kružno pohranjivanje** znači da se nakon što se blok **napuni**, sljedeći **bajt** opet puni od **početne** adrese.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; početak

              ORG   0
              B GLAVNI                     ; preskok u glavni


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; prekidni potprogram

              ORG   1C                     ; adresa za FIQ
PREKID        STMFD R13!, {LR}             ; spremanje konteksta (koriste se privatni
                                           ; registri R8 i R9 od moda FIQ)

              LDR   R8, RTC                ; dohvat adrese RTC-a
              MOVE  R9, #0
              STR   R9, [R8,#0C]           ; vrati brojilo na 0
              STR   R9, [R8,#08]           ; dojavi prihvat prekida

              LDR   R8, GPIO               ; dohvat adrese GPIO-a
              LDR   R9, [R8,#4]            ; očitaj sklopku sa vrata B
              ANDS  R9, R9, #80             ; ispitaj stanje sklopke
              BEQ   NIJE_SKLOPKA

              ; sklopka JE pritisnuta
JE_SKLOPKA    MOVE  R9, #0D               ; briši LCD
              BL    LCDWR
              MOVE  R9, #4F               ; slovo O
              BL    LCDWR
              MOVE  R9, #4E               ; slovo N
              BL    LCDWR
              MOVE  R9, #0A               ; prikaži ON na LCD-u
              BL    LCDWR
              B     VAN

              ; sklopka NIJE pritisnuta
NIJE_SKLOPKA  MOVE  R9, #0D               ; briši LCD
              BL    LCDWR
              MOVE  R9, #0A               ; prikaži prazan LCD
              BL    LCDWR

VAN           LDMFD R13!, {LR}            ; obnovi kontekst
              SUBS  PC, LR, #4            ; povratak iz FIQ


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; adrese i konstanta

GPIO          DW    0FFFF1000              ; adrese sklopova
RTC           DW    0FFFF2000
KONST         DW    %D 1000               ; konstanta brojenja
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; potprogram LCDWR

              ; parametri:   znak se prenosi registrom R9
              ;              adresa GPIO-a se prenosi registrom R8

LCDWR         STMFD R13!, {R9}      ; spremi kontekst

              BIC   R9, R9, #80    ; brišemo gornji bit za svaki slučaj - neobavezno
                                   ; (može i AND)
              STR   R9, [R8]         ; slanje znaka

              EOR   R9, R9, #80      ; dizanje impulsa (može i ORR)
              STR   R9, [R8]

              EOR   R9, R9, #80      ; spuštanje impulsa (može i BIC ili AND)
              STR   R9, [R8]

              LDMFD R13!, {R9}      ; obnovi kontekst
              MOV   PC, LR          ; povratak



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; glavni program

GLAVNI        MOV   SP, #10<12      ; inicijalizacija stoga

              ; inicijalizacija smjera GPIO-a
              LDR   R0, GPIO        ; dohvat adrese GPIO-a
              MOV   R1, #0FF        ; smjerovi = FF
              STR   R1, [R0,#8]     ; cijeli A = izlaz
              STR   R1, [R0,#0C]    ; cijeli B = ulaz

              ; inicijalizacija RTC-a
              LDR   R0, RTC               ; dohvat adrese RTC-a
              LDR   R1, KONST
              STR   R1, [R0, #04]          ; match register
              MOV   R1, #1
              STR   R1, [R0,#10]           ; kontrolna riječ: prekid = ON
              MOV   R1, #0
              STR   R1, [R0,#0C]         ; brojilo = 0 (neobavezno)

              MRS   R0, CPSR
              BIC   R0, R0, #40            ; dozvoli FIQ
              MSR   CPSR_c, R0

              LDR   R1, GPIO       ; pripremi adrese za petlju
              MOV   R2, #10<8        ; adresa bloka
              MOV   R3, #0         ; ofset u bloku

LOOP          LDR   R0, [R1,#4]     ; čitanje sa porta
              BIC   R0, R0, #80     ; pobriši bit sklopke
              STRB  R0, [R2, R3]    ; spremi u blok

              ADD   R3, R3, #1        ; povećavaj kružno R3
              CMP   R3, #10<4      ; provjera broja podataka i vraćanje sa 100 na 0
              MOVEQ R3, #0

              ; Može se R3 povećati za 1:    ADD  R3, R3, #1
              ; a zatim mu se briše bit 8:   BIC  R3, R3, #10<4
              ;
              ; Može i bez R3 tako da se provjerava je li adresa u R2 došla do 1100.
              ;
              ; Mogu i razni drugi načini.

     B     LOOP                     ; petlja se vrti beskonačno
```