Završni ispit iz Arhitekture računala 1 **21. lipnja 2018.**

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 bod) Podatak 0000110_2 u **7-bitnom NBC-u** predstavlja broj __6__, a u **7-bitnom formatu 2'k** predstavlja broj __6__. Podatak 1101_2 u **4-bitnom NBC-u** predstavlja broj __13__, a u **4-bitnom formatu 2'k** predstavlja __-3__. **OVDJE OBAVEZNO NAPIŠITE POSTUPAK:**

0000110 = 2+4 = 6 NBC i 2'k (isti prikaz za oba jer je broj pozitivan što se vidi iz najvišeg bita)
1101 = 8 + 4+ 1 = 13 NBC
1101 je negativan broj u 2'k, pa radimo dvojni komplement da dobijemo apsolutni iznos broja 0010+1=0011
0011= 2+1 = 3, rezultat je -3

1.b. (1,5 boda) Oduzmite u **5 bitova** brojeve 01001 - 10101. Rezultat oduzimanja je (binarno): __10100__, a zastavice će biti postavljene ovako: prijenos = __0__, posudba = __1__, ništica = __0__, predznak = __1__, preljev = __1__. **OVDJE OBAVEZNO NAPIŠITE POSTUPAK:**

```
 01011  - prijenosi      prijenos =0
  01001                  posuda = not prijenos = 1
  01010                  preljev =  0 xor 1 = 1
+     1                  predznak =   1
  10100                  ništica = 0 (jer rezultat nije 0)
```

1.c (1,5 bod) Memorijske i ulazno-izlazne sabirnice imaju različite karakteristike. U svakoj tvrdnji **zaokružite** onu sabirnicu za koju vrijedi dotična tvrdnja:

a) Memorijska **da** / ulazno-izlazna sabirnica ima veliku brzinu prijenosa podataka.
b) Memorijska **da** / ulazno-izlazna sabirnica ima malu duljinu.
c) Memorijske **da** / ulazno-izlazne sabirnice su obično sinkrone.
d) Memorijska / ulazno-izlazna **da** sabirnica je bolje prilagođena različitim brzinama rada.
e) ARM-ova sabirnica APB je memorijska / ulazno-izlazna **da**.
f) ARM-ova sabirnica AHB je memorijska **da** / ulazno-izlazna.

1.d. (1,5 bod) Napišite **smjerove** FRISC-ovih priključaka: READ je **izlazni**, WRITE je **izlazni**, ADR je **izlazni**, DATA je **dvosmjerni**, WAIT je **ulazni**, SIZE je **izlazni**.

1.e (1 bod) U mikroarhitekturi **FRISC**-a postoji sklop **shuffler**. **Zaokružite** naredbe za čije ispravno izvođenje je **nužan** shuffler: LOAD **ne**, STORE **ne**, LOADB **da** i STOREB **da**.

1.f. (1 bod) Za procesor **ARM** napišite po **dvije** naredbe koje imaju isti učinak na registre i memoriju kao i naredba:

(1) `STMFD SP!,{R0, R1}`

```
STR R1,[SP,#-4]!
STR R0,[SP,#-4]!
```

(2) `LDMFD SP!, {R0,R1}`

```
LDR R0,[SP],#4
LDR R1,[SP],#4
```

Ne smijete koristiti naredbe LDM i STM !

1.g. (0,5 boda) Procesor **ARM sa statičkim predviđanjem grananja** izvodi programski odsječak s desne strane. Za naredbu **BHI** predvidjet će se da: **se grananje neće dogoditi**.
Za naredbu **BNE** predvidjet će se da: **će se grananje dogoditi**.

```
LABELA1     CMP   ...
            BHI   LABELA2
...
            CMP   ...
LABELA2     BNE   LABELA1
```

1.h. (*2 bod*) Nakon uključenja **ARM7** izvodi ovaj programski odsječak. Uz svaku naredbu napišite koliko **puta** se izvodi i koliko **ciklusa** traje pojedino izvođenje (npr. 5 x 1c + 1 x 2c znači da naredba pet puta traje po jedan ciklus i jednom traje dva ciklusa).

```
        ORG 0
        MOV   R0, #7          1c + 2c punjenje
        LDR   R1, ADR         3c
LAB     SUBS  R0, R0, #1      7 x 1c
        BNE   LAB             6 x 3c + 1 x 1c
        STR   R0, [R1],#4     2c
```

Ukupno trajanje odsječka je 34 ciklusa

2. FRISC (9 bodova) Na FRISC su spojeni GPIO1 i GPIO2, CT i DMA-sklop. Adrese im odaberite sami.

FRISC **beskonačno** čita podatke sa GPIO1 koji radi kao **uvjetna** jedinica. Podatci se kao **bajtovi** spremaju u memorijski blok od adrese 1000_{16}.

Svakih **5 minuta** se do tada primljeni podatci iz bloka 1000_{16} **pomoću DMA** trebaju poslati na GPIO2 koji radi kao **bezuvjetna** jedinica. DMA radi **zaustavljanjem procesora**. **Nakon završenog DMA-prijenosa**, nastavljaju se čitati podatci iz GPIO1, ali se opet pune **od početka bloka**, tj. od adrese 1000_{16}. To se ponavlja **beskonačno**.

Kašnjenje od 5 minuta ostvarite sklopom **CT** na čiji ulaz dolazi signal frekvencije **100 Hz**. CT daje prekide na **NMI**.

Zbog jednostavnosti, pretpostavite da se unutar 5 minuta blok memorije neće prepuniti i da DMA-prijenos traje puno kraće od 5 minuta. Također zanemarite slučaj kad NMI dolazi upravo u trenutku spremanja podatka iz GPIO1 u memoriju.

U ovom zadatku je pogrešno pretpostavljeno da DMA-sklop može prenositi pojedinačne bajtove (iz bloka 1000 u GPIO). Zato "pravo" rješenje nije moguće. U rješenju ispod je napravljeno kao da DMA ne radi s 32-bitnim podatcima nego s bajtovima. Moglo bi se napraviti da se prenosi četverostruko manje podataka, ali ni to rješenje nije dobro jer GPIO ne može primati cijeli 32-bitni podatak (u rješenju ćemo priznavati sve varijante).

```
D_SRC       EQU   0FFFF0000
D_DST       EQU   0FFFF0004
D_N         EQU   0FFFF0008
D_CR        EQU   0FFFF000C
D_GO        EQU   0FFFF0010
D_BS        EQU   0FFFF0014

C_CR        EQU   0FFFF3000
C_LR        EQU   0FFFF3004
C_IACK      EQU   0FFFF3008
C_IEND      EQU   0FFFF300C

P1_CR       EQU   0FFFF4000
P1_D        EQU   0FFFF4004
P1_BS       EQU   0FFFF4008
P1_END      EQU   0FFFF400C

P2_CR       EQU   0FFFF5000
P2_D        EQU   0FFFF5004
P2_IACK     EQU   0FFFF5008          ; ova adresa nije nužna
P2_IEND     EQU   0FFFF500C          ; ova adresa nije nužna

            ORG   0                  ; neobavezno jer ORG 0 je default
            MOVE  10000, SP
            JP    GLAVNI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  adrese i početak

            ORG   0C             ; adresa prekidnog potprograma za NMI

            PUSH  R0               ; spremi kontekst (ovdje SR nije nužno spremiti
            MOVE  SR, R0         ; jer nije mijenjan, tj. nema ALU-naredaba)
            PUSH  R0

            STORE R0, (C_IACK)   ; dojava prihvata prekida

DMA_INIT    MOVE  1000, R0       ; izvorište podataka: memorijski blok na adresi 1000
            STORE R0, (D_SRC)
```

```
        MOVE  P2_D, R0      ; odredište podataka: adresa podatka na GPIO2
        STORE R0, (D_DST)

        MOVE  %B 1000, R0   ; kontrolna riječ: mem->vj, halting, bez prekida
        STORE R0, (D_CR)
        ; D_DST i D_CR se mogu inicijalizirati i u glavnom programu

        LOAD  R0, (BROJAC)  ; broj podataka za DMA-prijenos
        STORE R0, (D_N)

        STORE R0, (D_GO)    ; pokreni DMA-prijenos

        MOVE  0, R0         ; vrati brojač podataka na 0 (**)
        STORE R0, (BROJAC)

        MOVE  1000, R0      ; vrati adresu bloka na 1000
        STORE R0, (ADRESA)

        STORE R0, (D_BS)   ; brisanje bistabila u DMA (nije nužno, ali je lijepo ;)
        STORE R0, (C_IEND)  ; dojava kraja posluživanja

        POP   R0            ; obnovi kontekst
        MOVE  R0, SR
        POP   R0

        RETN                ; povratak iz NMI

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


GLAVNI  MOVE  %B 001, R0           ; inicijaliziraj GPIO1: ulazni, bez prekida
        STORE R0, (P1_CR)

        MOVE  %B 010, R0           ; inicijaliziraj GPIO2: postavljanje bitova
        STORE R0, (P2_CR)        ; u postavljanju bitova nema spremnosti ni prekida

        MOVE  %D 30000, R0         ; konstanta za 5min = 5*60*100 = 30000
        STORE R0, (C_LR)
        MOVE  %B 111, R0           ; inicijaliziraj CT: NMI, prekid, broji
        STORE R0, (C_CR)

        ; ovdje se djelomično može inicijalizirati DMA (destination i kontrolna riječ)

CEKAJ   LOAD  R0, (P1_BS)          ; čekaj spremnost GPIO1
        CMP   R0, 0
        JR_EQ CEKAJ

        LOAD  R0, (P1_D)          ; čitaj podatak sa GPIO1
        STORE R0, (P1_BS)         ; briši status bistabil od GPIO1

        LOAD  R1, (ADRESA)        ; spremi podatak na adresu iz ADRESA
        STOREB R0, (R1)
        ADD   R1, 1, R1           ; povećaj adresu u ADRESA
        STORE R1, (ADRESA)

        LOAD  R2, (BROJAC)        ; povećaj brojač podataka BROJAC (**)
        ADD   R2, 1, R2
        STORE R2, (BROJAC)

        STORE R0, (P1_END)        ; dojavi kraj posluživanja na GPIO1

        JR    CEKAJ               ; beskonačna petlja


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


ADRESA  DW    1000                ; trenutna adresa za upis podatka
BROJAC  DW    0                   ; brojač trenutno upisanih podataka
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  globalne varijable

Komentar: umjesto odvojenih varijabli za adresu i brojač, bolje je pamtiti samo adresu
ili samo brojač. Npr., iz adrese sa lako izračuna broj podataka. U glavnom programu i
prekidnom potprogramu ne bi bilo dijelova označenih sa (**).

U prekidnom potprogramu bi se umjesto LOAD R0,(BROJAC) napravilo:
          LOAD R0, (ADRESA)
          SUB  R0, 1000, R0
```

3. ARM (9 bodova) Napišite **potprogram F_X** koji računa $F(X) = X^2 + X/5$. **Parametar X** se prenosi pomoću **memorijske lokacije iza naredbe BL**, a rezultat se vraća **registrom R0**. X i rezultat su 32-bitni brojevi u formatu **NBC**. Pretpostavite da kod izračuna rezultata neće doći do prekoračenja opsega. Za množenje **nemojte** koristiti petlju (tj. metodu uzastopnog pribrajanja), a dijeljenje je **cjelobrojno**.

Napišite **potprogram OBLOK** koji obrađuje blok **32-bitnih NBC** podataka u memoriji. OBLOK ima dva 32-bitna parametra koji se **prenose stogom**. Jedan parametar zadaje **adresu bloka**, a drugi zadaje **broj podataka u bloku**. Potprogram OBLOK **nema** povratne vrijednosti. Potprogram OBLOK obrađuje blok tako da za **svaki podatak** u bloku **pozove F_X** i vraćenu vrijednost **F(X) upiše preko originalnog podatka** u bloku.

Glavni program treba **pomoću OBLOK** obraditi memorijski blok na **adresi 1000_{16}** u kojem ima **10_{16} podataka**.

```
             ; glavni program
GLAVNI       MOV   SP, #10<12      ; inicijalizacija stoga

             MOV   R1, #10<8       ; adresa prvog bloka 1000
             MOV   R2, #10         ; veličina prvog bloka 10
             STMFD SP!, {R1,R2}     ; parametri na stog
             BL    OBLOK           ; poziv potprograma
             ADD   SP, SP, #8      ; ukloni parametre sa stoga
             SWI   123456          ; haltanje


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

OBLOK        STMFD SP!, {R0,R1,R2,LR}     ; spremanje konteksta
                                          ; obavezno spremiti LR (tj. R14) i R0 jer
                                          ; njega mijenja potprogram F_X

             ADD   R0, SP, #%D16          ; računa se adresa parametara
             LDMFD R0, {R1, R2}           ; dohvat parametara:
                                          ; R1=adresa,  R2=veličina
; Iznad može i LDMI (isto što i LDMFD). Umjesto ADD i LDMFD može i ovako:
; LDR R1,[SP,#10], LDR R1,[SP,#14]

LOOP         LDR   R0, [R1]               ; dohvat podatka iz bloka

             STR   R0, PARAM              ; upis parametra na memorijsku lokaciju iza BL
POZIV        BL    F_X                    ; poziv potprograma F_X
PARAM        DW    0                      ; lokacija za slanje parametra u F_X
             STR   R0, [R1],#4            ; spremi rezultat u blok i povećaj adresu

             SUBS  R2, R2, #1             ; smanji brojač petlje
             BNE   LOOP                   ; uvjet za kraj petlje

             LDMFD SP!, {R0,R1,R2,LR}     ; obnova konteksta
             MOV   PC, LR                 ; povratak

; Učitavanje parametara na početku programa može i izravno bez izračuna adrese:
;     LDR  R1, [SP,#%D16]
;     LDR  R2, [SP,#%D20]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; potprogram OBLOK

F_X          STMFD SP!, {R1,R2}           ; spremanje konteksta
```

```
                LDR   R0, [LR],#4            ; dohvat parametra i pomicanje LR za 4 na
                                              ; pravu povratnu adresu

                MUL   R1, R0, R0             ; računaj X² i spremi u R1

                MOV   R2, #0                 ; rezultat dijeljenja R2 je početno 0
DIJELI          SUBS  R0, R0, #5             ; oduzmi X-5
                ADDHS R2, R2, #1             ; povećaj rezultat ako X>5 ili X==5
                BHI   DIJELI                 ; ponavljaj dok je X>5
                ; Iznad se koriste uvjeti HI i HS jer se radi sa NBC brojevima

                ADD   R0, R1, R2             ; računaj X²+X/5 i spremi u povratni registar

                LDMFD SP!, {R1,R2}           ; obnova konteksta
                MOV   PC, LR                 ; povratak

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; potprogram F_X
```

4. ARM (12 bodova) Na ARM su spojeni GPIO1, RTC1 i RTC2. Adrese im odaberite sami.

RTC1 je spojen na **FIQ**, a na ulaz mu dolazi signal frekvencije **1MHz**.

RTC2 je spojen na **IRQ**, a na ulaz mu dolaze impulsi koje treba prebrajati.

Na **vrata A** od GPIO spojen je uređaj koji **bezuvjetno** prima 8-bitne podatke.

Na **vrata B** od GPIO je spojen **temperaturni sklop kao na predavanjima** (*podsjetnik: bitovi 0-5 su iznos temperature; bit 6 je ulazni za dojavu valjanosti temperature, bit 7 je izlazni za dojavu da je temperatura pročitana*).

Svake stotinke (mjeriti vrijeme pomoću **RTC1**) treba **očitati temperaturu** i **spremiti je kao bajt** u memoriju u **blok** od adrese 300_{16}. Pri tome se **prebraja koliko** očitanih temperatura je spremljeno u memorijski blok.

Svaki puta kada **RTC2 odbroji 5000_{10} impulsa**, uređaju na **vratima A** treba poslati podatak o **broju** temperatura spremljenih u memorijskom bloku na adresi 300_{16}, a **daljnja mjerenja** opet treba početi puniti od **početka bloka**. Također se i **brojač** spremljenih temperatura treba resetirati **na nulu**.

Program radi **beskonačno**. Zbog jednostavnosti zanemarite slučaj istovremene pojave IRQ i FIQ.

```
                ORG   0                    ; neobavezno, jer ORG 0 je default
                B     GLAVNI                 ; preskok u glavni

                ORG   18                     ; početak prekidnog za IRQ
                B     IRQ

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; početak

                ORG   1C                     ; adresa za FIQ
                ; može i skok u potprogram, ali bolje ga je napisati izravno na 1C

                ; nema spremanja konteksta (koriste se privatni registri moda FIQ)
                ; ako se koriste registri R0-R7, obavezno ih treba spremiti

                ; reinicijalizacija RTC-a
FIQ             LDR   R8, RTC1               ; dohvat adrese RTC1
                STR   R9, [R8,#08]           ; dojavi prihvat prekida
                MOV   R9, #0
                STR   R9, [R8,#0C]           ; vrati brojilo na 0


                LDR   R8, GPIO               ; dohvat adrese GPIO-a
CEKAJ           LDR   R9, [R8,#4]            ; očitaj vrata B
                ANDS  R10, R9, #40             ; ispitaj stanje temperaturnog sklopa i...
                BEQ   CEKAJ                ; čekaj dok je bit_6 jednak nuli

                ; temperatura JESTE valjana
CITAJ           MOV   R10, #80               ; digni bit 7 (ili ORR R9,R9,#80 pa STR R9)
                STR   R10, [R8, #4]
                MOV   R10, #00               ; spusti bit 7 (ili AND R9,R9,#7F pa STR R9)
```

```
                STR   R10, [R8, #4]

SPREMI          AND   R9, R9, #3F           ; počisti sinkronizacijske bitove, tj.
                                           ; pripremi temperaturu za spremanje
                LDR   R10, BROJAC           ; pročitaj brojač spremljenih mjerenja
                STRB  R9, [R10, #300]       ; spremi izmjerenu temperaturu u memoriju
                ADD   R10, R10, #1          ; povećaj brojač
                STR   R10, BROJAC           ; spremi natrag brojač u memoriju

                SUBS  PC, LR, #4            ; povratak iz FIQ

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; prekidni potprogram za FIQ

IRQ             STMFD R13!, {R1,R2}         ; spremanje konteksta

                ; reinicijalizacija RTC-a
                LDR   R1, RTC2               ; dohvat adrese RTC2
                STR   R2, [R1,#08]           ; dojavi prihvat prekida
                MOV   R2, #0
                STR   R2, [R1,#0C]           ; vrati brojilo na 0


                LDR   R1, GPIO               ; dohvat adrese GPIO
                LDR   R2, BROJAC             ; dohvat brojača mjerenja
                STR   R2, [R1]               ; slanje brojača na vrata A
                MOV   R2, #0
                STR   R2, BROJAC             ; resetiranje brojača na 0

                LDMFD R13!, {R1,R2}         ; obnova konteksta
                SUBS  PC, LR, #4            ; povratak iz IRQ

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; prekidni potprogram za IRQ



GPIO            DW    0FFFF1000       ; adrese sklopova
RTC1            DW    0FFFF2000
RTC2            DW    0FFFF3000

KONST1          DW    %D 10000        ; konstanta brojenja za RTC1 (ne može u MOV)
KONST2          DW    %D 5000         ; konstanta brojenja za RTC2 (ne može u MOV)

BROJAC          DW    0               ; brojač izmjerenih temperatura

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; adrese, konstanta i brojač


                ; glavni program
GLAVNI          MOV   SP, #10<12      ; inicijalizacija stoga

                ; inicijalizacija smjera GPIO-a
                LDR   R0, GPIO        ; dohvat adrese GPIO-a

                MOV   R1, #0FF       ; vrata A - definiranje smjera
                STR   R1, [R0,#8]      ; A(0-7) = izlazni smjer

                MOV   R1, #07F       ; vrata B - definiranje smjera
                STR   R1, [R0,#0C]     ; B(0-6) = ulaz, B7 = izlaz

                ; inicijalizacija RTC1 (mjeri vrijeme)
                LDR   R0, RTC1              ; dohvat adrese RTC-a

                LDR   R1, KONST1           ; ne može MOV jer je konstanta "prevelika"
                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)
```

```
               ; inicijalizacija RTC2 (broji impulse)
               LDR   R0, RTC2              ; dohvat adrese RTC-a

               LDR   R1, KONST2           ; ne može MOV jer je konstanta "prevelika"
               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, #C0            ; dozvoli FIQ i IRQ (bitovi 6 i 7 u nulu)
               MSR   CPSR_c, R0

LOOP           B     LOOP                  ; petlja se vrti beskonačno

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; glavni program
```