Završni ispit iz Arhitekture računala 1R 26.1.2022.

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

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

Dozvoljeno je koristiti isključivo službene šalabahtere (popise naredaba) za procesor ARM i FRISC-V. Programe treba pisati uredno i komentirati pojedine cjeline. Ispit se piše 150 minuta.

TEORIJSKI DIO (Inačica A):

Sljedećih 22 pitanja odnosi se na teorijsko poznavanje arhitektura procesora ARM i FRISC-V. Na pitanja odgovarate zacrnjivanjem odgovarajućeg polja u **obrascu za odgovore**. Pri ispravljanju, boduju se **isključivo pitanja odgovorena na obrascu za odgovore**. Svako točno odgovoreno pitanje nosi **1 bod**, netočno odgovoreno pitanje nosi **-0.2 boda** (odbija se od cjelokupnog ispita), a neodgovoreno pitanje nosi **0 bodova**. Svako teorijsko pitanje ima **isključivo jedan točan odgovor**.

Općeniti teorijski dio (brojevni sustavi i kodovi, zapis podataka u memoriji, zastavice prilikom obavljanja ALU operacija na generičkom N-bitnom računalu): ~5 pitanja

Teorijski dio vezan uz procesor ARM: ~11 pitanja

Teorijski dio vezan uz procesor FRISC-V: ~6 pitanja

PROGRAMSKI ZADACI (1 FRISC-V, 1 ili 2 ARM)

1. (FRISC-V) U sustavu se nalaze se procesor **FRISC-V**, sklop **GPIO0** na adresi **FFFF1000$_{(16)}$** na čija **vrata A** je spojen temperaturni senzor, te sklop **GPIO1** na adresi **FFFF2000$_{(16)}$**, na čija **vrata A, bit 0** je spojen prekidač.

Sustav uzastopno mjeri temperaturu očitavanjem temperaturnog senzora sve dok je prekidač u stanju **logičke nule** te očitanja **kružno** zapisuje u blok memorije od adrese **500$_{(16)}$**. Kada program zapiše **20$_{(10)}$** vrijednosti, počinje kružno upisivati podatke od početka bloka, tj. ponovno od adrese **500$_{(16)}$**. Ovime se ostvaruje funkcionalnost tzv. kružnog međuspremnika (eng. *circular buffer*).

Kada prekidač pritiskom prijeđe u stanje **logičke jedinice**, program računa **prosječnu** vrijednost podataka iz kružnog međuspremnika (zadnjih **20$_{(10)}$** mjerenja) i zapisuje ju na adresu **1000$_{(16)}$**. Sustav prestaje očitavat s temperaturnog senzora do ponovnog pritiska tipke i prelaska u stanje **logičke 0**. Očitane vrijednosti se nastavljaju upisivati u kružni međuspremnik od pozicije na kojoj su stale.

Pretpostavite da postoji potprogram **DIJELI** koji prima argument **djeljenik** putem registra **a0**, **djelitelj** putem registra **a1**. **Rezultat dijeljenja** vraća registrom **a0**.

Možete pretpostaviti da je prije pokretanja programa u memoriji od adrese $500_{(16)}$ zapisano $20_{(10)}$ mjerenja s vrijednosti 0, te da su sve vrijednosti vrata oba GPIO sklopa inicijalno postavljene u stanje **logičke 0**.

```
STACK_POINTER_BASE_ADDR equ   0x10000
GPIO0_BASE_ADDR         equ   0xFFFF1000
GPIO1_BASE_ADDR         equ   0xFFFF2000

BUFFER_SIZE             equ   20

glavni  lui     sp, %hi(STACK_POINTER_BASE_ADDR)

        lui     s0, %hi(BUFFER_BASE_ADDR)
        addi    s0, s0, %lo(BUFFER_BASE_ADDR)

        addi    s1, s1, BUFFER_SIZE
```

```
        addi    s2, x0, 0       ; pokazivac

        lui     s3, %hi(GPIO0_BASE_ADDR)
        addi    s3, s3, %lo(GPIO0_BASE_ADDR)

        addi    t0, x0, 0b10000000
        sw      t0, 8(s3)

        lui     s4, %hi(GPIO1_BASE_ADDR)
        addi    s4, s4, %lo(GPIO1_BASE_ADDR)

        addi    t0, x0, 0b00000001
        sw      t0, 8(s4)

        addi    s5, x0, PROSJEK_ADDR

        ; set pin direction for GPIO1

petlja  lbu     t0, 0(s4)       ; Provjeri prekidač
        andi    t0, t0, 0b00000001
        beq     t0, x0, temp

        addi    a0, s0, 0
        addi    a1, s1, 0
        addi    a2, s3, 4
        jal     ra, prosjek
        sw      a0, 0(s5)

cekaj_prekidac
        lbu     t0, 0(s4)
        andi    t0, t0, 0b00000001
        bne     t0, x0, cekaj_prekidac

temp    addi    a0, s3, 0
        jal     ra, citaj_temp

        add     t0, s0, s2      ; izracunaj adresu za zapis u buffer
        sb      a0, 0(t0)

        addi    s2, s2, 1       ; povecaj brojac zapisanih podataka

        addi    t0, s2, -BUFFER_SIZE
        blt     t0, x0, dalje
        addi    s2, x0, 0

dalje   jal     x0, petlja

; citaj_temp
;
; Cita vrijednost temperaturnog senzora.
;   * a0 - adresa temperaturnog senzora
; Povratna vrijendost:
;   * a0 - vrijednost s temperaturnog senzora
citaj_temp
        addi    sp, sp, -8
        sw      t0, 0(sp)
        sw      t1, 4(sp)
```

```
cekaj_rdy
        lbu     t0, 0(a0)
        andi    t1, t0, 0b01000000
        beq     t1, x0, cekaj_rdy

        ; Impuls na ACK signal
        ori     t0, t0, 0b10000000
        sw      t0, 0(a0)
        andi    t0, t0, 0b01111111
        sw      t0, 0(a0)

        andi    a0, t0, 0b00111111

        lw      t0, 0(sp)
        lw      t1, 4(sp)
        addi    sp, sp, 8
        jalr    x0, 0(ra)

; ispis_prosjek
;
; Ispisi prosječnu vrijednost temperature iz cirkularnog buffera.
;   * a0 - početna adresa buffera
;   * a1 - broj podataka u bufferu
; Rezultat
;   * a0 - prosjek
prosjek
        addi    sp, sp, -16
        sw      t0, 0(sp)
        sw      t1, 4(sp)
        sw      t2, 8(sp)
        sw      ra, 12(sp)

        addi    t0, x0, 0       ; suma
        addi    t1, a1, 0

petlja_prosjek
        lbu     t2, 0(t0)

        add     t0, t0, t2

        addi    a0, a0, 1
        addi    t1, t1, -1
        bne     t1, x0, petlja_prosjek

        addi    a0, t0, 0       ; a0 <- suma
        ; od prije je broj podataka u a1
        jal     ra, dijeli

        lw      t0, 0(sp)
        lw      t1, 4(sp)
        lw      t2, 8(sp)
        lw      ra, 12(sp)
        addi    sp, sp, 16
        jalr    x0, 0(ra)
```

```
                ORG   0x500
BUFFER_BASE_ADDR
        DW 0, 0, 0, 0, 0
        DW 0, 0, 0, 0, 0
        DW 0, 0, 0, 0, 0
        DW 0, 0, 0, 0, 0

                ORG     0x1000
PROSJEK_ADDR    DW      0
```

2. (ARM) Za procesor ARM napišite potprogram PODIJELI koji **cjelobrojno dijeli** (zanemaruje decimalna mjesta) dva **32-bitna broja u formatu 2'k** (**Paziti na predznak!**). Potprogram prima dva 32-bitna parametra **preko stoga**. Prvi parametar koji pozivatelj stavlja na stog je djeljenik, a drugi djelitelj. Rezultat dijeljenja treba vratiti **registrom R0**. Dijeljenje možete ostvariti tako da prebrajate koliko puta djelitelj stane u djeljenik (**uzastopno oduzimanje ili zbrajanje**). **Nije potrebno pisati glavni program**.

```
PODIJELI    STMFD SP!, {R1, R2, R3} ;

            ; Inicijalizacija registara
            ADD R0, SP, #%D 12      ; Alternativa: LDR Rx, [(SP|R13), #%D(16|12)]
            LDMFD R0, {R1, R2}      ; R1, R2 := djeljenik, djelitelj
                                    ; Nacin adresiranja nije bitan (moze FD),
                                    ; bitno je samo da se ucitaju dva parametra sa
stoga.
            MOV R0, 0               ;
                                    ;    R0 := rezultat
            MOV R3, 0               ;    R3 := predznak

            CMP R1, 0               ; Ako je djeljenik negativan, pretvori u poz.
            EORMI R1, R1, -1        ;
            ADDMI R1, R1, 1         ;
                                    ;    Alternativa: RSBMI R2, R2, #0
            EORMI R3, R3, 1         ; Pamti predznak

            CMP R2, 0               ;
            BEQ KRAJ                ;  Ako je djelitelj jednak nuli,
                                    ;    izadji odmah van
            EORMI R2, R2, -1        ;
            ADDMI R2, R2, 1         ;
            EORMI R3, R3, 1         ; Pamti predznak

            ; Uzastopno oduzimanje
PETLJA      SUBS R1, R1, R2         ; Mora imati "S" za postavljanje zastavica
            ADDGE R0, R0, 1         ; Dobar uvjet, može i HS(unsigned),ali samo ako
            BGE PETLJA              ; pretvore oba operanda u pozitivne

            CMP R3, 1               ; Provjera finalnog predznaka
            EOREQ R0, R0, -1        ;
            ADDEQ R0, R0, 1         ;

KRAJ        LDMFD SP!, {R1, R2, R3} ;
            MOV PC, LR              ;
```

3. (ARM) Na ARM su spojeni **GPIO** i **RTC** (adrese odaberite sami). Na RTC je spojen signal od **100 Hz**, 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 bitove od 0 do 3 spojene su tipke (daju '1' kada su pritisnute).
Napišite program za upravljanje digitalnim satom. Digitalni sat prikazuje sate (0-23) i minute (0-59) na LCD prikazniku u formatu SS:MM (ASCII kodovi: '0'=30 (nula), ':'=3A, LCD briši=0D, LCD prikaži=0A). Za mjerenje protjecanja vremena koristite sklop RTC, a za izračun ASCII vrijednosti desetica (podijeliti s 10) potrebno je koristiti **potprogram PODIJELI** iz prethodnog zadatka (**Nije potrebno prepisivati!**).
Dodatno, potrebno je ostvariti funkcionalnost postavljanja sata. Sat se postavlja pomoću 4 tipke spojene na **vrata B GPIO** sklopa:

- bit 0 dodaje +1 sat
- bit 1 oduzima -1 sat
- bit 2 dodaje +1 minutu
- bit 3 oduzima -1 minutu

Nakon pritiska bilo koje tipke, program mora čekati da se sve tipke vrate u **početno stanje** (vrijednost '0') prije nego što krene očitavati sljedeći pritisak na tipku. Tijekom postavljanja, sat ne prestaje raditi. Napišite potprogram LCDWR (kao na predavanjima) pomoću kojega ćete slati pojedini znak na LCD (parametar se prenosi putem registra R9). Inicijalnu vrijednost sata je 00:00.

```
                ORG 0
                B GLAVNI                ;

                ORG 0x1C                ; FIQ - RTC
                B PREKIDNI
;------------ početak  -----------------------------

GLAVNI          MSR   CPSR, #0b11010001  ; prelazak u način rada FIQ

                MOV   R13,  #0x10000     ; inicijalizacija R13_fiq
                MSR   CPSR, #0b11010011  ; prelazak u način rada SVC
                MOV   R13,  #0xFC00      ; inicijalizacija R13_svc

                LDR R0, RTC             ;
                LDR R1, KONST           ;
                STR R1, [R0, #4]        ; postavi match registar

                MOV R1, #1
                STR R1, [R0, #0x10]           ; omogući prekid

                MRS R0, CPSR
                BIC R0, R0, #0x40       ; omogući FIQ
                MSR CPSR_c, R0

                LDR R0, GPIO            ;
                MOV R1, #0xFF           ; svi bitovi su izlazni
                STR R1, [R0, #8]

                MOV R1, #0F             ; bitovi 0-3 su ulazni
                STR R1, [R0, #0x0C]

                BL ISPIS                ; inicijalni ispis

      ; (može na više načina ispitivati)
                ; poolanje
                ; mijenjane vrijednosti s tipkama
POOL            LDR R1, [R0, #4]  ; ispitaj tipke
                ANDS R1, R1, #0x0F  ; Moze i TST R1, #0F
```

```
                BEQ POOL

                ANDS R2, R1, #1   ; plus 1 sat
                BNE PLUSAT

                ANDS R2, R1, #2   ; minus 1 sat
                BNE MINSAT

                ANDS R2, R1, #4   ; plus 1 minuta
                BNE PLUMIN

MINMIN          LDR R1, MINS      ; ucitaj i
                CMP R1, #0        ; provijeri opseg
                MOVLE R1, #59     ; može i EQ
                SUBGT R1, #1      ;
                STR R1, MINS      ; spremi i idi na otpusti
                B DALJE           ;

PLUSAT          LDR R1, HRS       ;
                CMP R1, #23 ;
                MOVGE R1, #0      ; može i EQ
                ADDLT R1, #1      ;
                STR R1, HRS       ;
                B DALJE           ;

MINSAT          LDR R1, HRS       ;
                CMP R1, #0        ;
                MOVLE R1, #23     ; može i EQ
                SUBGT R1, #1      ;
                STR R1, HRS       ;
                B DALJE           ;

PLUMIN          LDR R1, MINS      ;
                CMP R1, #59 ;
                MOVGE R1, #0      ; može i EQ
                ADDLT R1, #1      ;
                STR R1, MINS      ;

DALJE           BL ISPIS          ; Ispisati novu vrijednost na ekran

OTPUSTI         LDR R1, [R0, #4]  ; čekaj da se sve tipke otpuste
                ANDS R1, R1, #0x0F
                BNE OTPUSTI

                B POOL            ;

;----------- glavni program  ------------------------

                GPIO DW 0xFFFF1000      ; adrese jedinica (proizvoljno)
                RTC DW 0xFFFF2000

                KONST DW 6000     ; 60 sekundi
                MINS DW 0         ;   stanje sata
                HRS  DW 0

;----------- adrese i konstante --------------------
; parametri: znak se prenosi registrom R9
; adresa GPIO-a se prenosi registrom R8
```

```
LCDWR           STMFD R13!, {R9}        ; spremi kontekst

                BIC   R9, R9, #0x80           ; brišemo gornji bit(neobavezno)
                STR   R9, [R8]          ; slanje znaka

                EOR   R9, R9, #0x80           ; dizanje impulsa (može i ORR)
                STR   R9, [R8]

                EOR   R9, R9, #0x80           ; spuštanje impulsa (BIC, AND)
                STR   R9, [R8]

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

ISPIS           STMFD R13!, {R0, R1, R2, R8, R9, LR} ; spremi kontekst

                LDR R8, GPIO            ;

                MOV R9, #0x0D                 ; briši LCD
                BL LCDWR

;========= priprema za ispis

                LDR R1, HRS             ; učitaj sate

                MOV R2, #10       ;
                                        ; postoji implementacija bez dijeljenja
                                        ; onda se desetice i jedinice
                                        ;     zapisuju u odvojenim varijablama
                                        ; onda prebaciti na dio gdje se
                                        ;     jedinice i desetice inkrementiraju
                STMFD R13!, {R1, R2}    ; može biti u potprogramu
                BL PODIJELI             ; podijeli s 10
                ADD R13, R13, #8        ; počisti stog

                ADD R0, R0, #0x30             ; dodaj za ASCII znak '0'

                MOV R9, R0              ; desetice (sati)
                BL LCDWR

                MUL R0, R0, R2          ; * 10
                SUB R0, R1, R0          ; ostatak

                MOV R9, R0              ; jedinice (sati)
                BL LCDWR
;=========== ispis sati

                MOV R9, #0x3A                 ; ':'
                BL LCDWR

                LDR R1, MINS            ; učitaj minute

                                        ; isto kao i za sate
                STMFD R13!, {R1, R2}    ; može biti u potprogramu
                BL PODIJELI             ; podijeli s 10
                ADD R13, R13, #8        ; počisti stog

                ADD R0, R0, #0x30             ; dodaj za ASCII znak '0'
```

```
                MOV R9, R0              ; desetice (minute)
                BL LCDWR

                MUL R0, R0, R2          ; * 10
                SUB R0, R1, R0          ; ostatak

                MOV R9, R0              ; jedinice (minute)
                BL LCDWR

;=========== ispis minuta

                MOV R9, #0x0A                ; prikaži
                BL LCDWR

                LDMFD R13!, {R0, R1, R2, R8, R9, LR} ; vrati kontekst
                MOV PC, LR              ;

;------------- potprogram ISPIS --------------------

PREKIDNI        STMFD R13!, {R0, R1, LR} ;

                LDR R0, RTC             ;
                MOV R1, #0
                STR R1, [R0, #0x0C]           ; resetiraj LR
                STR R1, [R0, #8]        ;  i dojavi prekid

                LDR R0, HRS             ;  učitaj sate i minute
                LDR R1, MINS

                CMP R1, #59             ; ispitaj minute
                ADDLT R1, R1, #1        ;  ako je manje od 59, dodaj 1
                STRLT R1, MINS          ;  i spremi
                BLT KRAJ

                MOV R1, #0              ; inače minute u 0
                ADD R0, #1              ;  a sati +1

                CMP R0, #24       ;
                MOVEQ R0, #0

                ŞTR R0, HRS             ; i spremi
                STR R1, MINS

KRAJ            BL ISPIS                ; ispisi novo stanje

                LDMFD R13!, {R0, R1, LR};
                SUBS PC, LR, #4         ;

;------------- prekidni potprogram --------------------
```