Međuispit iz Arhitekture računala 1R **16. 11. 2021.**

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 službeni šalabahter (popis naredaba) za procesor ARM. Programe treba pisati uredno i komentirati pojedine cjeline. Ispit se piše 90 minuta.

TEORIJSKI DIO (Inačica A):

Sljedećih 16 pitanja odnosi se na teorijsko poznavanje arhitekture procesora ARM. 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**.

Sljedeća 4 pitanja (1. – 4.) vezana su uz blok memorije:

adresa	vrijednost
103	14
102	74
101	9E
100	A2

1. Heksadecimalna vrijednost **16-bitnog little-endian** podatka na adresi **100**:
   a. A29E
   b. 9EA2
   c. 2AE9
   d. E92A
   e. Niti jedan od navedenih
2. Heksadecimalna vrijednost **16-bitnog big-endian** podatka na adresi **102**:
   a. 7414
   b. 4741
   c. 1474
   d. 749E
   e. Niti jedan od navedenih
3. Vrijednost **8-bitnog 2'k** podatka zapisanog na adresi **100**:
   a. $94_{(10)}$
   b. $-94_{(10)}$
   c. $-93_{(10)}$
   d. $162_{(10)}$
   e. $95_{(10)}$
4. Vrijednost **8-bitnog NBC** podatka zapisanog na adresi **101**:
   a. $98_{(10)}$
   b. $-98_{(10)}$
   c. $-97_{(10)}$
   d. $158_{(10)}$
   e. $97_{(10)}$

U sljedeća tri pitanja (5. - 7.) razmatramo odlomak koda za procesor **ARM**.

```
1:          B SKOK
2: SEST     DW 6
3: SKOK     LDR R6, SEST
4:          MOV R0, #18
5: PETLJA   SUBS R0, R0, R6
6:          MOV R6, R6, LSL #1
7:          BPL PETLJA
8:          STR R0, REZ
```

5. Koliko ciklusa traje izvođenje odlomka koda? (Pretpostavite da je odlomak dio nekog većeg koda pa cikluse dohvata i dekodiranja prve naredbe B SKOK ne treba računati)
   a. 22
   b. 25
   c. 28
   d. 29
   e. Ništa od navedenog
6. Na kojim linijama se događa upravljački hazard?
   a. 1 i 7
   b. 1 i 3
   c. 3 i 8
   d. 1 i 8
   e. 1 i 6
7. Na kojim linijama se događa strukturni hazard?
   a. 1 i 7
   b. 1 i 3
   c. 3 i 8
   d. 1 i 8
   e. 1 i 6
8. Koju naredbu treba iskoristiti za povrat konteksta ako se iskoristila naredba STMED za spremanje?
   a. LDMED
   b. LDMFA
   c. LDMDB
   d. STMED
   e. Niti jedna od navedenih
9. Za što se (po dogovoru) koriste registri R13 i R14 na procesoru **ARM**?
   a. R13 - pokazivač vrha stoga, R14 - programsko brojilo
   b. R13 - povratna adresa za vraćanje iz potprograma, R14 - programsko brojilo
   c. R13 - povratna adresa za vraćanje iz potprograma, R14 - pokazivač vrha stoga
   d. R13 - pokazivač vrha stoga, R14 - registar za pohranu povratne adrese
   e. R13 - programsko brojilo, R14 - pokazivač vrha stoga
10. U **ARM**-ovom registru R5 je broj 200_{16}. U memoriji, počevši od adrese 200_{16}, redom su upisani bajtovi: 11_{16}, 22_{16}, 33_{16}, 44_{16}, 55_{16}, 66_{16}, 77_{16}, 88_{16}. Što se nalazi u registrima R0 i R5 nakon izvršavanja naredbe:

    ```
    LDRB R0, [R5, #4]!
    ```

    a. R0 = 44; R5 = 200
    b. R0 = 88; R5 = 204
    c. R0 = 88; R5 = 200
    d. R0 = 11; R5 = 200
    e. R0 = 55; R5 = 204

1. (X bodova) 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**. Pozivatelj stavlja na stog djeljenik na nižu adresu a djelitelj na višu adresu. Stog je tipa FD. 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**. U slučaju dijeljenja s nulom vratite rezultat=0.

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

            ; Inicijalizacija registara
            ADD R0, SP, #12         ; Adresa za učitavanje parametara.
            LDMFD R0, {R1, R2}      ; R1 = djeljenik, R2= djelitelj

            MOV R0, #0               ; R0 -> Rezultat
            MOV R3, #0               ; R3 -> Predznak

            CMP R1, #0               ; Ako je djeljenik negativan,
            MVNMI R1, R1        ; pretvori ga u pozitivan.
            ADDMI R1, R1, #1

            EORMI R3, R3, #1         ; Zapamti predznak rezultata.

            CMP R2, #0               ; Ako je djelitelj jednak nuli,
            BEQ KRAJ                ; izadji odmah van.

            MVNMI R2, R2        ; Ako je djeljitelj negativan,
            ADDMI R2, R2, #1         ; pretvori ga u pozitivan.
            EORMI R3, R3, #1         ; Zapamti predznak rezultata

            ; Uzastopno oduzimanje
PETLJA      SUBS R1, R1, R2         ; 'S' za postavljanje zastavica.
            ADDGE R0, R0, #1         ; Dok je djeljenik veći od nule,
            BHI PETLJA              ; dodaj 1 u rezultat.

            CMP R3, #1               ; Ako rezultat treba biti negativan,
            MVNEQ R0, R0        ; pretvori ga u 2'k negativan broj.
            ADDEQ R0, R0, #1

KRAJ        LDMFD SP!, {R1, R2, R3} ; Obnovi kontekst.
            MOV PC, LR              ; Povratak iz potprograma.
```

2. (X bodova) Za procesor ARM napišite potprogram PARITET koji preko lokacije iza naredbe BL prima 32-bitni podatak i provjerava mu paritet. Ako mu je paritet neparan, treba preko R0 vratiti broj 1, a ako je paran, treba preko R0 vratiti 0.

Napisati potprogram OBRADI koji obrađuje blok 32bitnih podataka tako da im prvo provjerava paritet, a zatim podatke s neparnim paritetom zamjenjuje podatkom $F0F0F0F0_{16}$. Potprogram OBRADI prima početnu adresu bloka podataka kao parametar i to preko stoga. Blok uvijek ima 100_{16} podataka. Potprogram OBRADI ne vraća rezultat. Potprogram OBRADI mora provjeravati paritetnosti pomoću potprograma PARITET.

U glavnom programu treba pomoću potprograma OBRADI obraditi blok podataka na adresi 1000_{16}. Svi potprogrami moraju čuvati vrijednost registara, a parametre sa stoga moraju uklanjati pozivatelji.

```
            ORG 0             ; glavni program ovaj org I ne mora
            MOV SP, #0x10000  ; inicijalizacija stoga

            MOV R7, #0x1000   ; adresa bloka 1000
            STMFD SP!, {R7}   ; spremanje parametra na stog

            BL OBRADI         ; poziv potprograma
            ADD SP, SP, #4    ; uklanjanje parametra

KRAJ        SWI 123456



OBRADI      STMFD SP!, {R0, R1, R2, R3, LR}     ; spremi kontekst i LR

            LDR R1, [SP,#20]  ; učitavanje parametra: R1=adresa bloka
            MOV R2, #0x100          ; brojač za petlju

            LDR R3, PODATAK         ; podatak za zamjenu

LOOP        LDR R0, [R1]            ; učitaj 32-bitni broj iz bloka

            STR R0, PARAM     ; pa ga stavi kao parametar iza naredbe BL

            BL PARITET        ; odredi paritet pomoću potprograma PARITET
PARAM       DW 0              ; mjesto za parametar

; brojeve s neparnim paritetom treba zamijeniti sa F0F0F0F0

            CMP R0, #1        ; ispitaj rezultat potprograma PARITET
            BNE DALJE
ZAMIJENI    STR R3, [R1]      ; upiši novi podatak u blok

; umjesto BNE+STR može samo STREQ R3, [R1]
DALJE       ADD R1, R1, #4    ; pomak adrese u bloku
```

```
            SUBS R2, R2, #1   ; smanjenje brojača petlje
            BNE LOOP

            LDMFD SP!, {R0, R1, R2, R3, LR}     ; obnova konteksta
            MOV PC, LR        ; povratak iz potprograma

; podatak se ne može se izravno napisati u ALU-naredbama
PODATAK     DW 0xF0F0F0F0


PARITET     STMFD SP!, {R1,R2}      ; spremi kontekst
            LDR R2, [LR], #4        ; učitaj parametar...
                                    ; ...i pomakni povratnu adresu

            MOV R0, #0              ; početna vrijednost brojača jedinica
            MOV R1, #32             ; brojač za petlju

LOOP2       MOVS  R2, R2, LSR #1    ; najniži bit u zastavicu C
            ADDCS R0, R0, #1        ; C==1 --> povećaj brojač jedinica
                                    ; može i bezuvjetno zbrajanje
                                    ; s prijenosom: ADC R0, R0, #0

            SUBS R1, R1, #1         ; smanji brojač petlje i ponavljaj
            BNE LOOP2

            AND R0, R0, #1          ; pretvori brojač jedinica u rezultat

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