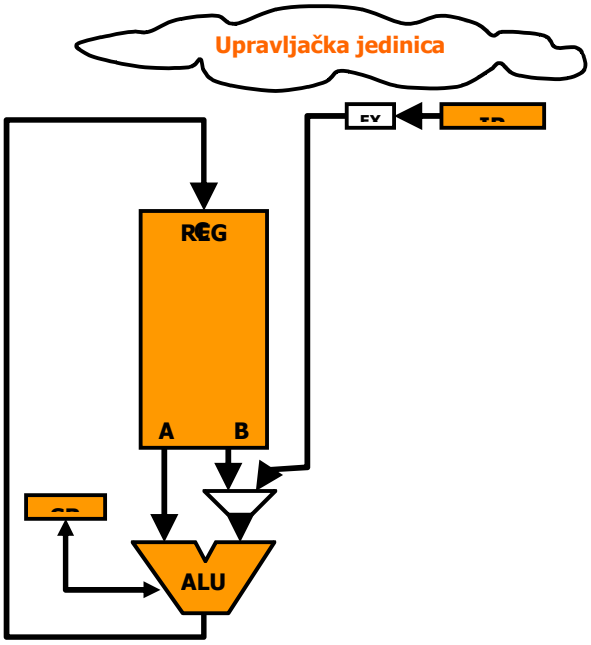
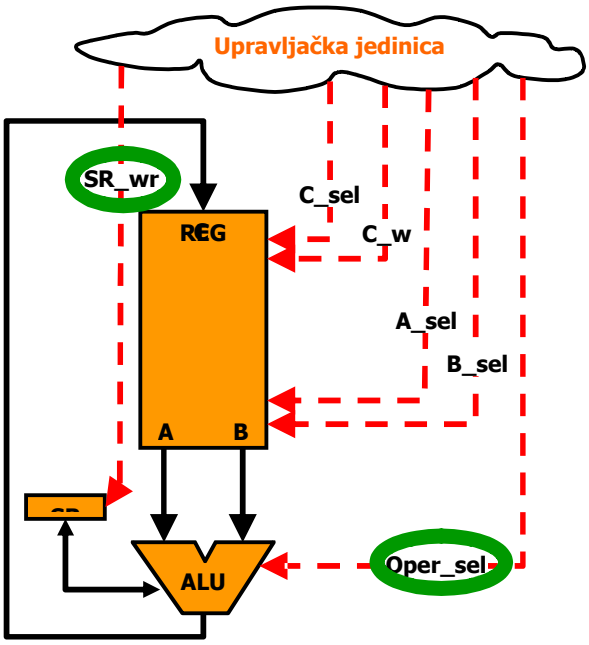
Rješenja prvog međuispita iz Arhitekture računala 1 (2005/2006)

1. (3 boda) Na donjim slikama djelomične arhitekture procesora FRISC nacrtajte:

a) (1,5 bod) Sve **potrebne podatkovne veze** koje omogućuju izvođenje aritmetičko-logičke naredbe (npr. ADD R0,R0,R0 i ADD R0,5,R0)

b) (1,5 bod) **Osnovne upravljačke signale** kojima upravljačka jedinica omogućuje izvođenje neke aritmetičko-logičke naredbe koja koristi dva registra kao ulazne operande.





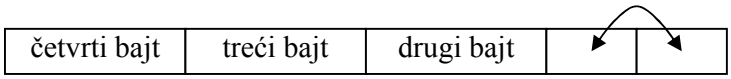
2. (6 bodova) U memoriji sustava s procesorom FRISC nalazi se blok 32-bitnih podataka u formatu s bitom za predznak (bit za predznak+31 bit podatka). Blok podataka počinje s podatkom koji se nalazi na adresi 1000(16) a završava sa podatkom koji se nalazi na adresi 1100(16). Napisati program za FRISC koji će zbrojiti sve podatke iz bloka. Zbroj mora biti u 64-bitnom formatu dvojnog komplementa. Zbroj treba spremiti od adrese 2000(16) u big endiannu.

Podatci zauzimaju 104 bajtova (prvi podatak počinje na 1000, a zadnji na 1100), pa je u bloku ukupno 104/4 = 41 32-bitni podatak (oprez: svi ovi brojevi su heksadekadski).

```
        MOVE 0, R6    ; mjesto za zbroj - niži dio
        MOVE 0, R7    ; mjesto za zbroj - viši dio

        MOVE 1000, R0    ; pokazivač podataka
        MOVE 41, R1      ; brojač za petlju

        LOAD R5,(MASKA) ; maska za brisanje bita predznaka

PETLJA  LOAD R3,(R0)     ; učitaj broj
        OR   R3,R3,R3    ; postavi zastavicu N
        JR_P POZ         ; ispitaj preznak broja

NEG     AND R3,R5,R3     ; ako je broj negativan
        XOR R3,-1,R3     ; pobriši mu bit preznaka
        ADD R3,1,R3      ; i pretvori ga u 2'k

        ADD R3,R6,R6     ; zbroji niži dio broja
        ADC R7,-1,R7     ; zboji viši dio broja i to
                         ; predznačno proširen sa FFFFFFFF

        JP NEXT          ; nastavi sa sljedećim brojem

POZ     ADD R3,R6,R6     ; ako je bbroj pozitivan
        ADC R7,0,R7      ; zbroji niži dio broja i viši dio
                         ; i to predznačno proširen sa 00000000

NEXT    ADD R0,4,R0      ; kraj petlje: povećanje pokazivača
        SUB R1,1,R1      ; smanjivanje brojača petlje
        JP_NZ PETLJA

KRAJ    ; Kraj postupka, treba spremiti broj u big endiannu

        ROTL R7, 8, R7        ; spremanje viših bajtova
        STOREB R7, (2000)     ; na niže adrese
        ROTL R7, 8, R7
        STOREB R7, (2001)
        ROTL R7, 8, R7
        STOREB R7, (2002)
        ROTL R7, 8, R7
        STOREB R7, (2003)

        ROTL R6, 8, R6       ; spremanje nižih bajtova
        STOREB R6, (2004)    ; na više adrese
        ROTL R6, 8, R6
        STOREB R6, (2005)
        ROTL R6, 8, R6
        STOREB R6, (2006)
        ROTL R6, 8, R6
```

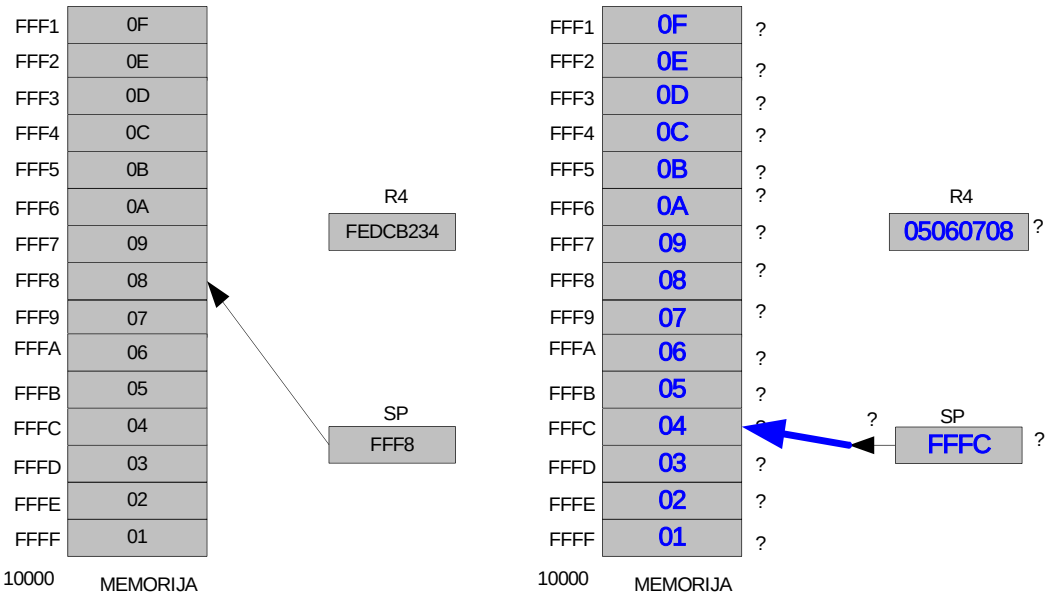
```
        STOREB R6, (2007)

        HALT

MASKA   DW    7FFF FFFF  ; maska za brisanje bita predznaka
```

3. (6 bodova) U memoriji od adrese 1000(16) nalazi se blok sa 32-bitnim podacima u formatu dvojnog komplementa. Blok je nepoznate duljine, ali se zna da je zaključen podatkom F000 0000. Napišite program za FRISC koji treba u svim pozitivnim podacima u bloku zamijeniti dvije najniže skupine od 4 bita (vidi sliku). Negativni podaci se ne mijenjaju.




```
      LOAD  R0, (ZADNJI)   ; učitavanje oznake kraja
      MOVE  1000, R1       ; pokazivač na podatke u bloku

LOOP  LOAD  R2, (R1)       ; učitaj podatak iz bloka

      CMP   R0, R2         ; ispitaj je li to zadnji podatak
      JR_EQ KRAJ           ; ako jeste, idi na kraj (HALT)

      OR    R2, R2, R2     ; postavi zastavice na temelju učitanog
      JR_N  NEXT           ; podatka i ispitaj mu predznak
                           ; ako je predznak negativan,
                           ; idi na sljedeći podatak

      ; broj je pozitivan - treba mu zamijeniti donja dva nibla

POZ   AND   R2, 0000 000F, R4    ; U R4 stavi prvi nibl
      AND   R2, 0000 00F0, R5    ; U R5 stavi drugi nibl

      ROTL  R4, 4, R4            ; Pomakni ih tako da
      ROTR  R5, 4, R5            ; zamijene mjesta

      AND   R2, 0FFFF FF00, R2   ; Pobriši donji bajt originalnog
                                 ; podatka

      OR    R2, R4, R2           ; Upiši u originalni podatak
      OR    R2, R5, R2           ; oba nibla

      STORE R2, (R1)             ; Prepiši preko starog podatka

NEXT  ADD   R1, 4, R1      ; Kraj petlje: povečaj pokazivač
      JP    LOOP           ; vrati se na početak petlje

KRAJ HALT

ZADNJI  DW   F000 0000    ; oznaka kraja - služi za usporedbu
```

1. a) zadatak (1,5 bod): Lijevi dio slike prikazuje početno stanje memorije i registara. Na desnoj strani slike upišite stanje memorije i registara (u za to predviđene kućice) nakon izvođenja naredbe *POP R4*:



1. b) zadatak (1,5 bod): Na crte upišite naziv procesorskog adresiranja koji je podvučen u sljedećim instrukcijama:

a) LOAD R5, (R0 + 4) _____**registarsko indirektno s odmakom**_____

b) JP_CC (R0) _____**registarsko indirektno**_____

c) JR 1200 _____**relativno**_____

d) JP 1200 _____**apsolutno**_____

e) ADD R0, %D 1, R2 _____**neposredno**_____

f) ADD R1, 34, R5 _____**registarsko**_____

Zadatak 1. rješavate na ovom papiru i predajete zajedno sa rješenjima zadataka 2. i 3. u košuljici. Međuispit traje 75 minuta.

2. zadatak (6 bodova): U memoriji računala s procesorom FRISC nalazi se blok 16-bitnih podataka u formatu dvojnog komplementa. Početna adresa bloka zapisana je od memorijske adrese 1000_{16} u 4 memorijske lokacije. Veličina bloka podataka zapisana je od memorijske adrese 1004_{16} u 4 sljedeće memorijske lokacije.

Napisati program koji će izračunati 32-bitni zbroj svih podataka iz bloka koji počinje od adrese 30000_{16} i ima 250_{16} podataka. Nakon toga, taj zbroj treba pretvoriti u 32-bitni format s bitom za predznak i spremiti ga u memoriju od adrese 1008_{16}. Pretpostavite da pri zbrajanju podataka neće doći do prekoračenja opsega.

```
      LOAD R0, (1000)   ; R0 je pointer na blok
      LOAD R1, (1004)   ; R1 je brojač za petlju
      MOVE 0, R2        ; u R2 će biti suma

LOOP  LOADH R3, (R0)    ; učitaj 16-bitni podatak (u 2'k)

      ROTL  R3, %D 16, R3     ; predznačno proširivanje
      ASHR  R3, %D 16, R3     ; sa 16 na 32 bita

; može se ispitivati 16. bit pa onda sa OR setirati gornjih 16 bitova

      ADD R3, R2, R2    ; zbrajanje proširenog podatka u sumu

      ADD R0, 2, R0     ; pomak pointera za 2 bajta
      SUB R1, 1, R1     ; smanjivanje brojača za petlju
      JR_NZ LOOP        ; ispitivanje kraja petlje


BPZ   ; pretvori sumu u R2 iz 2'k u format s bitom za predznak

      OR R2, R2, R2     ; postavi zastavicu N
      JR_P POZIT        ; ako je pozitivan, ne treba pretvarati

NEGAT ; ako je negativan, treba ga pretvoriti

      XOR R2, -1, R2    ; računanje apsolutne vrijednosti
      ADD R2, 1, R2

      ROTL R2, 1, R2    ; postavljanje predznaka u najviši bit
      OR R2, 1, R2
      ROTR R2, 1, R2
; može i OR sa maskom učitanom iz memorije

POZIT STORE R2, (1008)  ; spremanje rezultata
      HALT

      `ORG 1000
      DW 30000    ; adresa bloka
      DW 250      ; broj podataka u bloku
      DW 0        ; mjesto za rezultat
```

3. zadatak (6 bodova): Napisati potprogram AVGB za FRISC koji treba izračunati srednju vrijednost **četiri** 8-bitnih NBC podataka. Podatci se prenose u potprogram preko stoga kao **jedan** 32-bitni podatak, gdje je **svaki bajt** jedan 8-bitni NBC podatak. Srednju vrijednost treba vratiti kao 32-bitni rezultat preko registra R0. Zbrajanje bajtova treba riješiti **petljom**.

Napisati glavni program koji za 8-bitne NBC podatke: 3_{16}, 1_{16}, $4F_{16}$ i 5_{16} računa srednju vrijednost i sprema je na lokaciju REZ. Potprogram mora čuvati vrijednosti registara, a pozivatelj treba ukloniti parametar sa stoga.

```
;;;; GLAVNI PROGRAM ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

GLAVNI     MOVE 10000, SP  ; inicijlizacija stoga

           LOAD R0, (PODATAK) ; Dohvat podatka, stavljanje na stog. Ne može sa MOVE.
           PUSH R0

           CALL AVGB      ; poziv

           ADD SP, 4, SP     ; ciscenje stoga i spremanje rezultata
           STORE R0, (REZ)

           HALT


PODATAK DW 03014F05    ; ili:    PODATAK DB 3,1,4F,5    ; podatci
REZ     DW 0


;;;; POTPROGRAM ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AVGB        PUSH  R1            ; pohrana konteksta
            PUSH  R2
            PUSH  R3

            LOAD  R1, (SP+%D16) ; ucitaj podatak

            ; inicijalizacija petlje i brojaca
            MOVE 4, R2 ; brojac za petlju
            MOVE 0, R0 ; suma

PETLJA      AND  R1, 0FF, R3 ; maskiraj oktet
            ROTR R1, 8, R1   ; pomak podatka (može i SHR)

            ADD  R3, R0, R0  ; dodaj sumi

            SUB R2, 1, R2    ; smanjivanje brojača i petlja
            JP_NZ PETLJA

; umjesto maskiranja i rotiranja, može se u petlji čitati izravno bajt po bajt za LOADB


DALJE       SHR R0, 2, R0    ; dijeljenje sa 4

            POP   R3    ; vracanje konteksta
            POP   R2
            POP   R1

            RET         ; povratak
```

Zadatak 1. rješavate na ovom papiru i predajete zajedno sa rješenjima zadataka 2. i 3. u košuljici. Međuispit traje 75 minuta.

1. Međuispit iz Arhitekture računala 1 27. ožujka 2008.

Prezime i ime (štampanim slovima):_____

Matični broj: _____ Grupa na predavanjima: _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Zadatke 1., 2., 3. i 4. rješavate na ovaj papir.

1. (1 *bod*) Koji realni broj **X** predstavlja podatak $\mathbf{42100000_{16}}$ koji je u 32-bitnom IEEE-formatu? **X** iznosi __**36**_____. (prikažite ga u dekadskoj bazi, a mora se vidjeti postupak pretvaranja na praznini ispod ovog zadatka).

ovdje bi trebao biti postupak pretvorbe...

2. (*1 bod*) Za opću ALU (nije vezano za procesor FRISC), na crte upišite ime zastavice i očekivano stanje zastavice (npr. prijenos=1) ako se detektira:

a) greška kod zbrajanja NBC-brojeva _____**PRIJENOS=1**_____

b) greška kod oduzimanja NBC brojeva _____**PRIJENOS=0 (ili POSUDBA=1)**_____

c) greška kod zbrajanja 2'k brojeva _____**PRELJEV=1**_____

d) greška kod oduzimanja 2'k brojeva_____**PRELJEV=1**_____

3. (*2 boda*) Prilikom dekodiranja FRISC-ove naredbe "*JP 100*", adresa na koju treba skočiti nalazi se u nižih 20 bitova registra __**IR**__. Prilikom izvođenja, ova 20-bitna adresa se prvo **_predznačno proširiti** _____ na širinu od __**32** __ bita, a nakon toga se upisuje u registar ___**PC**___. Naredba "*JP 100*" koristi procesorsko adresiranje koje se naziva _____**apsolutno adresiranje**_____.

4. (*2 boda*) Prije izvođenja FRISC-ove naredbe *POP*, stanje registra *SP* bilo je $\mathbf{2000_{16}}$. Nakon izvođenja naredbe *POP* u *SP* će biti vrijednost ___$\mathbf{2004_{16}}$____. Kao registar *SP*, zapravo se koristi registar ___**R7** ____. Prilikom izvođenja naredbe "*POP R2*" se adresa iz *SP* treba upisati u registar ___**AR** __, a podatak koji je dohvaćen sa stoga mora "proći" kroz registar ___**DR** ____, prije nego se upiše u *R2*.

5. (*3 boda*) Za procesor FRISC napisati program koji računa srednju vrijednost podataka u bloku memorije. U bloku se nalazi 64 podatka koji su u 32-bitnom formatu 2'k. Blok je zapisan od početne adrese $\mathbf{500_{16}}$. Srednju vrijednost potrebno je zapisati na lokaciju $\mathbf{400_{16}}$. **NAPOMENA**: suma podataka u bloku ne prelazi 32-bitni opseg.

```
`ORG 0
     MOVE %D64, R0   ; inicijalizacija brojača i adrese
     MOVE 500, R1
     MOVE 0, R4      ; suma,(može i XOR R4,R4,R4 ili AND R4,0,R4 itd.)

PETLJA
     LOAD R3,(R1)    ; učitavanje podatka
     ADD  R3, R4, R4 ; zbrajanje
     ADD  R1, 4, R1  ; sljedeća memorijska lokacija
     SUB  R0, 1, R0  ; smanjujemo brojač podataka
     JR_NZ  PETLJA   ; ispitujemo da li smo zbrojili svih 64

     ASHR  R4, 6, R4 ; dijeljenje sa 64 - 2'k format
     STORE R4,(400)  ; spremanje na adresu 400
     HALT            ; zaustavljanje procesora
```

6. (*6 bodova*) Za procesor FRISC napisati potprogram PRETVORI koji pretvara 32-bitni podatak iz formata s bitom za predznak u 32-bitni format 2'k. Podatak za pretvorbu prenosi se u potprogram putem stoga, a rezultat pretvorbe vraća se registrom R0. Potprogram treba čuvati stanja registara.

Napisati glavni program koji traži veći od dva 32-bitna podatka. Podaci su pohranjeni u memoriji na lokacijama $\mathbf{100_{16}}$ i $\mathbf{104_{16}}$ u formatu s bitom za predznak i treba ih pretvoriti u format 2'k korištenjem potprograma PRETVORI. Pretvorene podatke treba pohraniti u memoriju na iste lokacije. Nakon toga treba usporediti podatke i strogo veći od njih pohraniti na lokaciju $\mathbf{12345678_{16}}$ u formatu 2'k.

```
; potprogram
PRETVORI
     PUSH R1         ; spremanje konteksta

     LOAD R0,(SP+8) ; učitavanje podatka, BITAN je ispravan odmak

     ROTL R0,1,R1    ; provjeravanje predznaka, (ili OR R0,R0,R0 pa JR_P itd.)
     JR_NC POZ

NEG  SHL R0,1,R0     ; brisanje predznaka
     SHR R0,1,R0

     XOR R0,0FFFFFFFF, R0 ; dvojni komplement
     ADD R0,1,R0

POZ  POP R1          ; za pozitivni: samo vrati broj i obnovi kontekst

     RET             ; povratak


-----------------------------

; glavni program
GLAVNI
     `ORG 0
     MOVE 10000,SP ; init stoga

     LOAD R1,(100)     ; učitavanje prvog podataka
     PUSH R1
     CALL PRETVORI
     POP R1            ; može i: ADD SP, 4, SP
     STORE R0, (100)   ; spremi prvi pretvoreni natrag u memoriju
```

```
      MOVE R0, R1       ; spremi ga i u R1

      LOAD R2,(104)     ; učitavanje drugog podataka
      PUSH R2
      CALL PRETVORI
      POP R2            ; može i: ADD SP, 4, SP
      STORE R0, (104)   ; spremi drugi pretvoreni natrag u memoriju

      LOAD R3,(LOK)     ; učitavanje adrese

      CMP R1,R0        ; usporedi
      JR_SGT VECI      ; uvjet: strogo veći

NIJE  STORE R0,(R3)    ; spremanje i zaustavljanje
      HALT

VECI  STORE R1,(R3)    ; spremi veći i zaustavi
      HALT             ; zaustavi procesor

LOK   DW 12345678      ; adresa rezultata


; ovo dolje nije trebalo pisati, ali nek se nađe u rješenju zbog potpunosti

      `ORG  100
PRVI  DW  neki_prvi_broj
DRUGI DW  neki_drugi_broj

      `ORG 12345678
MAX    DW  0           ; mjesto za rezultat
```

1. Međuispit iz Arhitekture računala 1 01. travnja 2009.

Prezime i ime (tiskanim slovima):_____

Matični broj: _____ Grupa na predavanjima: _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve zadatke rješavate na ovaj papir. Međuispit traje 75 minuta.

1. *(2 boda)* Brojeve 35 i -35 prikažite u odgovarajućim oblicima (formatima) i upišite rješenja u sljedeću tablicu:

	Broj	
Oblik	**35**	**-35**
8-bitni NBC	00100011	/ (nije moguće prikazati)
8-bitni prikaz 1'k	**(nije potrebno prikazati)**	11011100
8 bitni prikaz 2'k	00100011	11011101
8-bitni pakirani BCD	00110101	**(nije potrebno prikazati)**
8-bitni prikaz s bitom za predznak	00100011	10100011

2. (*2 boda*) Prilikom dekodiranja i izvođenja FRISC-ove naredbe "*ADD R0, 30, R1*", broj 30 se nalazi u 20 nižih bitova registra ____IR_____. Naredba koristi dva adresiranja koja se zovu __registarsko_____ i _neposredno / immediate _____. Prilikom izvođenja naredbe, na jedan ulaz ALU dovodi se podatak iz _____EXT____(nije bitan redoslijed EXT i R0)__, a na drugi se ulaz dovodi podatak iz _____R0_____ (napisati iz kojih dijelova procesora se dovodi podatak).

3. (*2 boda*) FRISC izvodi sljedeći program:

```
      `ORG 5
      MOVE   100, SP
      CALL   POTP
      HALT
POTP  RET
```

Upišite u tablicu s desna stanje svih memorijskih lokacija nakon izvođenja gornjeg programa (početno su sve prikazane memorijske lokacije lokacije u 0).

adresa	sadržaj memorijske lokacije (heksadekadski)
F8	00
F9	00
FA	00
FB	00
FC	10
FD	00
FE	00
FF	00
100	00
101	00
102	00
103	00

4. (*3,5 bodova*) Napišite potprogram koji prima dva ulazna parametra preko stoga (nazovimo ih A i B). Parametar A je onaj kojeg pozivatelj prvoga stavlja na stog, a parametar B je onaj kojega stavlja drugoga. Parametri su 32-bitni NBC brojevi. Potprogram treba izračunati vrijednost izraza 128_{10}*A+B i vratiti rezultat preko registra *R0*. Pretpostavite da neće doći do prekoračenja opsega od 32 bita. Potprogram treba čuvati stanja registara, a parametre treba uklanjati pozivatelj. Glavni program treba pozvati potprogram pri čemu kao parametar A treba poslati vrijednost s memorijske lokacije 1000_{16}, a kao parametar B podatak s memorijske lokacije 50000_{16}. Rezultat treba spremiti na lokaciju $1F00_{16}$.

```
POTP  PUSH  R1               ; spremi kontekst
      LOAD  R0, (SP+0C)      ; prvi (A)
      LOAD  R1, (SP+8)       ; drugi (B)
      SHL   R0, 7, R0        ; R0=128*A
      ADD   R0, R1, R0       ; R0=128*A+B
      POP   R1               ; vrati kontekst
      RET                    ; povratak



      `ORG 0
GLAVNI
      MOVE  10000, SP        ; stog na vrhu memorije
      LOAD  R0, (1000)       ; podatak s lokacije 1000
      PUSH  R0               ; na stog kao A
      LOAD  R0, (50000)      ; podatak s lokacije 50000
      PUSH  R0               ; na stog kao B
      CALL  POTP             ; pozovi potprogram
      ADD   SP, 8, SP        ; ocisti stog

      STORE R0, (1F00)       ; spremi rezultat
      HALT
```

5. (*5,5 bodova*) Napišite program koji čita 32-bitovne podatke iz memorije počevši od lokacije 1000_{16}, sve dok ne pročita oznaku kraja – podatak 12345678_{16}. Svaki od pročitanih podataka treba zamijeniti s novom 32-bitovnom vrijednošću koja se dobije na sljedeći način: Svaki od 4 bajta pročitanog podatka sadrži broj u zapisu 2'k (svaki broj je širine 8 bita). Ta četiri broja treba zbrojiti u 32-bitovnu vrijednost, te rezultat pohraniti u memoriju umjesto pripadnog originalnog podataka. Oznaku kraja nije potrebno zamijeniti s novom vrijednošću.

Primjer: u memoriji na lokaciji 1020_{16} nalazi se podatak $010203FF_{16}$. Nakon izvođenja programa na lokaciji 1020_{16} bit će upisana vrijednost 00000005_{16} = 1+2+3+(-1). Analogno za ostale memorijske lokacije.

```
        `ORG 0
GLAVNI  LOAD    R1, (OZNAKA)   ; ucitaj oznaku kraja
        MOVE    1000, R0       ; pocetak bloka

PETLJA  LOAD    R2, (R0)       ; ucitaj podatak
        CMP     R1, R2         ; provjeri oznaku kraja
        JP_EQ   KRAJ

ZBROJ   LOADB   R2, (R0)       ; ucitaj prvi bajt
        SHL     R2, %D24, R2   ; pomakni lijevo
        ASHR    R2, %D24, R2   ; predznacno prosiri

        LOADB   R3, (R0+1)     ; ucitaj drugi bajt
        SHL     R3, %D24, R3
        ASHR    R3, %D24, R3
        ADD     R2, R3, R2     ; zbroji

        LOADB   R3, (R0+2)
        SHL     R3, %D24, R3
        ASHR    R3, %D24, R3
        ADD     R2, R3, R2     ; zbroji

        LOADB   R3, (R0+3)
        SHL     R3, %D24, R3
        ASHR    R3, %D24, R3
        ADD     R2, R3, R2     ; zbroji

        STORE   R2, (R0)       ; spremanje zbroja
        ADD     R0, 4, R0      ; pomak na sljedeću adr.
        JP      PETLJA         ; nastavak petlje

KRAJ    HALT


OZNAKA  DW      12345678
```

1. međuispit iz Arhitekture računala 1 — 02. travnja 2010.

Grupa na predavanjima: _____

Prezime i ime (velikim 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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve zadatke rješavati na ovaj papir. Međuispit traje 75 minuta.

1. *(2 boda)* Brojeve 52_{10} i -52_{10} upišite u odgovarajućim oblicima (formatima) u sljedeću tablicu:

Oblik	Broj 52_{10}	Broj -52_{10}
8-bitni NBC	0011 0100	nije moguće
8-bitni prikaz 1'k	0011 0100	1100 1011
8 bitni prikaz 2'k	0011 0100	1100 1100
8-bitni pakirani BCD	0101 0010	nije moguće
8-bitni prikaz s bitom za predznak	0011 0100	1011 0100

2. (*2 boda*) Smjerovi FRISC-ovih priključaka su: READ je _ izlazni __, WRITE je _____ izlazni __, WAIT je ____ulazni_, ADR je _izlazni _, DATA je _ulazno/izlazni (dvosmjerni)_. Čitanje iz brze memorije traje _ 1_ takt(ova) CLOCK-a, a pisanje traje _ 1_ takt(ova). Pisanje u sporu memoriju traje __ 2 ili više; (1+TW, TW>1)___ takt(ova). Ako je memorija spora, to javlja FRISC-u preko sabirničke linije spojene na njegov priključak ____WAIT_____, kojega FRISC ispituje ____ na padajući brid signala CLOCK _____(u kojem trenutku).

3. (*2 boda*) FRISC izvodi sljedeći program:

```
      `ORG 0
      MOVE   100, SP
      PUSH   1234
      CALL   POTP
      ADD    SP, 8, SP
      PUSH   0ABCD
      HALT
POTP  PUSH   10
      RET
```

Upišite u tablicu desno stanje svih memorijskih lokacija od F0 do 107 i strelicom označite položaj SP nakon izvođenja gornjeg programa.
Početno su sve prikazane memorijske lokacije u 0.
Pretpostavite da PUSH radi sa src2.

Adresa	Sadržaj
F0	00
F1	00
F2	00
F3	00
-> F4	CD
F5	AB
F6	00
F7	00
F8	0C
F9	00
FA	00
FB	00

Adresa	Sadržaj
FC	34
FD	12
FE	00
FF	00
100	00
101	00
102	00
103	00
104	00
105	00
106	00
107	00

4. (*3,7 bodova*) Napišite potprogram POTP koji množi dva broja uzastopnim zbrajanjem. Glavni program šalje dva parametra potprogramu tako da prvi parametar šalje preko stoga, a drugi parametar putem R0. Parametri su 32-bitni brojevi u formatu NBC. Potprogram vraća rezultat u registru R2. Potprogram čuva stanja registara, a parametre sa stoga uklanja glavni program.
Napišite glavni program koji poziva potprogram POTP pri čemu kao prvi parametar šalje podatak zapisan na memorijskoj lokaciji 20000_{16}, a kao drugi parametar podatak na memorijskoj lokaciji 20004_{16}. Rezultat izvođenja potprograma treba spremiti na lokaciju 56781234_{16}.

5. (*5,3 bodova*) U memoriji se nalazi blok koji se sastoji od parova 16-bitnih podataka u prikazu s bitom za predznak. Memorijski blok počinje na adresi 2000_{16} , a zaključen je parom podataka $F0F0_{16}$ i $F0F0_{16}$.
Napišite program koji svaki par 16-bitnih brojeva u bloku memorije zamjenjuje novim 32-bitnim podatkom na sljedeći način. Za svaki par 16-bitnih brojeva u bloku treba izračunati njihov zbroj. Zbroj treba biti u 32-bitnom prikazu 2'k i treba ga zapisati u blok na mjesto početnog para brojeva.
Prilikom pretvorbe podataka, program također treba prebrajati koliko je negativnih 16-bitnih podataka bilo u početnom bloku i to na kraju programa treba zapisati na memorijsku lokaciju BROJAC.
Primjer (zbrajanje za jedan par podataka): U memoriji se na adresi 2000_{16} nalazi podatak 8005_{16}, a na adresi 2002_{16} 0002_{16}, što su brojevi -5 i +2. Ovaj par treba pretvoriti u broj $-3_{10} = -5_{10} + 2_{10} = FFFFFFFD_{16}$, koji treba zapisati na adresu 2000_{16}, a brojač treba povećati za 1.

Rješenje 4. zadatka:

```
                                ; potprogram
       `ORG 100                 ; ORG <> 0 ili poredak
POTP   PUSH   R1                ; spremi kontekst
       PUSH   R0
       MOVE   0, R2             ; postavi rezultat
       CMP    R0, 0             ; pogledaj da nije 0 u R0
       JR_EQ  GOTOV
       LOAD   R1, (SP+0C)       ; učitaj prvi
PETLJA ADD    R1, R2, R2        ; zbroji
       SUB    R0, 1, R0         ; oduzmi broj ponavljanja
       JP_NE  PETLJA            ; ponavljaj ako nije
GOTOV  POP    R0
       POP    R1                ; vrati kontekst
       RET                      ; povratak
                                ; glavni
       `ORG 0                   ; poredak ili ORG 0
GLAVNI MOVE   10000, SP         ; stog na vrhu memorije
       LOAD   R0, (20000)       ; učitaj podatak s lokacije
       PUSH   R0                ; stavi na stog
       LOAD   R0, (20004)       ; učitaj podatak s lokacije
       CALL   POTP              ; pozovi potprogram
       ADD    SP, 4, SP         ; očisti stog
       LOAD   R1, (MEM)         ; učitaj mem. lokaciju
       STORE  R2, (R1)          ; spremi rezultat
       HALT
MEM    DW 56781234              ; mem. adresa rezultata
       `ORG 20000
       DW 0ABCD1234, 0CDEF3412  ; neki podaci
```

Rješenje 5. zadatka:

```
       `ORG 0
GLAVNI MOVE   2000, R0     ; početak bloka
       MOVE   0, R1        ; brojač negativnih
PETLJA LOADH  R2, (R0)     ; učitaj prvi 16-bitni u R2
       LOADH  R3, (R0+2)   ; učitaj drugi 16-bitni u R3
       CMP    R2, 0F0F0    ; provjeri da li F0F0
       JP_NE  DALJE        ; ako ne skoči na oduzmi
       CMP    R3, 0F0F0    ; provjeri da li F0F0
       JP_EQ  KRAJ         ; skoči na kraj
DALJE  AND    R2, 8000, R5 ; pogledaj 16. bit
       JP_Z   DRUGI        ; ako pozitivan skok
       ADD    R1, 1, R1    ; povećaj brojač
       AND    R2, 7FFF, R2 ; brišem 16. bit
       XOR    R2, -1, R2   ; komplement
       ADD    R2, 1, R2    ; dodam 1 za 2k'
DRUGI  AND    R3, 8000, R5 ; pogledaj 16. bit
       JP_Z   ZBROJI       ; ako pozitivan skok
       ADD    R1, 1, R1    ; povećaj brojač
       AND    R3, 7FFF, R3 ; brišem 16. bit
       XOR    R3, -1, R3   ; komplement
       ADD    R3, 1, R3    ; dodam 1 za 2k'
ZBROJI ADD    R3, R2, R2   ; zbroji
       STORE  R2, (R0)     ; spremanje
       ADD    R0, 4, R0    ; pomak za 4
       JP     PETLJA       ; nastavak petlje
KRAJ   STORE  R1, (BROJAC) ; spremanje brojača
       HALT
BROJAC DW     0
```

1. međuispit iz Arhitekture računala 1 — 28. ožujka 2011.

Grupa na predavanjima: _____

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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Međuispit traje 75 minuta.

1. *(1 bod)* 4-bitna aritmetičko-logička jedinica oduzima binarne brojeve 0111 - 1100. Nakon operacije će biti: prijenos = **_0_** , posudba = **_1_** , preljev = **_1_** , ništica = **_0_** , predznak = **_1_** . Potrebno je napisati postupak rješenja.

2. (1,5 *bodova*) FRISC izvodi sljedeći program od adrese 0.

```
        `ORG 0
        LOAD R0, (ADR)
        LOAD R2, (R0+4)
        MOVE DALJE, R1
        JP   DALJE
ADR     DW   0C
DALJE   HALT
```

Nakon zaustavljanja procesora, u registrima će se nalaziti vrijednosti (zapisano **heksadekadski**):

R0 = _____**0C**_____

R1= _____**14**_____

R2 = _____**0C**_____

3. (1 bod) U programu iz prethodnog zadatka, odredite koje se vrste **procesorskog** adresiranja koriste.
U naredbi `MOVE` koriste se dvije vrste adresiranja: ____**registarsko**_____ i ____**neposredno**_____.
U naredbi `JP` koristi se _____**apsolutno**_____.
Adresiranje u **drugom** operandu naredbe "`LOAD R0,(ADR)`" naziva se ____**apsolutno**_____.
Adresiranje u **drugom** operandu naredbe "`LOAD R2,(R0+4)`" naziva se **_registarsko indirektno s odmakom (offsetom)__**

4. (1 bod) Upišite u memoriju, od adrese 1000, podatak 12345678_{16}. Širina svake memorijske lokacije je 8 bita.

	u redoslijedu little-endian
1000	**78**
1001	**56**
1002	**34**
1003	**12**

	u redoslijedu big-endian
1000	**12**
1001	**34**
1002	**56**
1003	**78**

5. (*5 bodova*) U memoriji se nalazi blok 16-bitnih brojeva u prikazu s bitom za predznak. Adresa početka bloka zapisana je u memorijskoj lokaciji s adresom 1000_{16}. Blok podataka završava podatkom 8000_{16} (ovaj podatak se ne smatra brojem u bloku).
Napišite program koji pretvara 16-bitne brojeve u 32-bitne brojeve i sprema ih istim redoslijedom u novi blok, počevši od adrese 5000_{16}. Brojeve treba pretvarati na sljedeći način. Program svaki **negativni** broj treba pretvoriti u 32-bitni broj u prikazu s bitom za predznak. Umjesto svakog **pozitivnog** broja, program u novi blok treba zapisati 32-bitni broj 0.
Novi blok treba zaključiti brojem 80000000_{16}. Prilikom pretvorbe brojeva, program također treba prebrajati koliko je **parnih** 16-bitnih brojeva bilo u početnom bloku i to treba zapisati na memorijsku lokaciju PARNI.

```
      `ORG 0
      LOAD R0, (1000)       ; R0 - početak izvorišnog bloka
      MOVE 5000, R3         ; R3 – početak odredišnog bloka
      MOVE 1, R5            ; maska za postavljanje 31. bita
      ROTR R5, 1, R5        ; ILI LOAD .... DW 80000000
      MOVE 0, R6            ; R6 - brojač parnih brojeva

POC   LOADH R1, (R0)        ; učitavamo 16-bitne brojeve
      CMP R1, 8000          ; ako je učitani broj 8000...
      JR_EQ KRAJ            ; ...pročitali smo cijeli blok

PREDZ AND R1, 8000, R2      ; ispitujemo bit za predznak
      JR_NZ NEG
POZ   MOVE 0, R2            ; pozitivne brojeve brišemo
      JR PARN               ; i skačemo na provjeru parnosti
NEG   AND R1, 07FFF, R2     ; negativnim brojevima brišemo 15.bit
      OR R2, R5, R2         ; i postavljamo 31. bit pomoću R5-maske

PARN  SHR R1, 1, R1         ; provjeravamo parnost
      JR_C SPREMI
      ADD R6, 1, R6         ; još jedan parni broj

SPREMI
      STORE R2, (R3)        ; spremamo broj u odredište
      ADD R0, 2, R0         ; povećavamo adresu 16-bitnog broja
      ADD R3, 4, R3         ; i adresu za 32-bitni novi broj
      JR POC                ; povratak na učitavanje

KRAJ  STORE R6, (PARNI)     ; na lok. PARNI stavljamo broj parnih
      MOVE 1, R7            ; zaključujemo blok s 80000000
      ROTR R7, 1, R7        ; ili LOAD + DW 80000000
      STORE R7, (R3)
      HALT
```

6. (*5,5 bodova*) Napišite potprogram NAJMANJI koji u bloku 32-bitnih brojeva u prikazu 2'k pronalazi **najmanji** broj. Glavni program šalje dva parametra potprogramu: adresu početka bloka šalje preko stoga, a broj 32-bitnih brojeva u bloku preko registra R0. Potprogram preko registra R1 vraća iznos najmanjeg broja u bloku. Potprogram treba čuvati stanja registara. Parametre sa stoga treba ukloniti glavni program.

Napišite glavni program koji treba pozvati potprogram NAJMANJI za blok na adresi 4000_{16} koji sadrži 100_{16} podataka. Nakon izvođenja potprograma, glavni program treba **spremiti vrijednost najmanjeg broja** na memorijsku lokaciju zadanu labelom MIN, koja se u memoriji nalazi odmah iza glavnog programa. Također, glavni program treba **obrisati** (postaviti na vrijednost 0) **sve podatke u bloku osim** onih koji su **jednaki najmanjem** broju.

```
      `ORG 0
GLAVNI
      MOVE 10000, SP        ; inicijalizacija stoga
      MOVE 4000, R2         ; R2 – početak bloka
      PUSH R2               ; stavljamo adresu početka bloka na stog
      MOVE 100, R0          ; R0 - broj podataka
      CALL NAJMANJI         ; pozivamo potprogram
      ADD SP, 4, SP         ; čistimo stog, 1 parametar
      STORE R1, (MIN)       ; spremamo najmanji na adresu MIN

POC   LOAD R6, (R2)         ; učitavamo podatak za usporedbu
      CMP R1, R6            ; preskačemo iste (najmanje) podatke
      JR_EQ PRESKOK         ;
      MOVE 0, R3            ; ako nisu isti, stavljamo 0 na mjesto broja
      STORE R3, (R2)        ; i spremamo na isto mjesto
PRESKOK
      ADD R2, 4, R2         ; povećavamo adresu za sljedeći broj
      SUB R0, 1, R0         ; smanjujemo brojač i vrtimo petlju
      JR_NZ POC
      HALT
MIN   DW 0
-----------------------
NAJMANJI
      PUSH R0
      PUSH R2
      PUSH R3
      PUSH R4

      CMP R0, 0             ; ako je blok veličine 0, idemo van da se
      JP_EQ KRAJ            ; ne vrtimo u beskonačnost…

      LOAD R2, (SP+14)      ; R2 – početak bloka
      LOAD R1, (R2)         ; R1 – najmanji broj (proglašavamo zasad prvi)

PETLJA
      LOAD R3, (R2)         ; trenutni podatak (za prvi prolaz bit će isti)
      CMP R3, R1            ; usporedba je li manji od najmanjeg?
      JR_SGT R3_VECI        ; Signed! 2'k brojevi
R3_MANJI
      MOVE R3, R1           ; ako je manji, postaje najmanji
R3_VECI
      ADD R2, 4, R2         ; pomičemo adresu za novi broj
      SUB R0, 1, R0         ; smanjujemo brojač i vrtimo petlju
      JR_NZ PETLJA
KRAJ  POP R4
      POP R3
      POP R2
      POP R0
      RET
```

Međuispit iz Arhitekture računala 1 **20. travnja 2012.** **Grupa na predavanjima: _____**

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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Međuispit traje 120 minuta.

1 a) (*3 boda*) FRISC izvodi sljedeći program:

```
        `ORG 0
        MOVE  100, SP
        MOVE  0FFFFABCD, R0
        PUSH   R0
        CALL    POTP
        ADD    SP, 8, SP
        CMP    R7,100
        HALT_NZ
        PUSH   R0
        HALT
POTP    MOVE   14, R0
        PUSH   R0
        RET
```

Adresa	Sadržaj
F0	
F1	
F2	
F3	
F4	
F5	
F6	
F7	
F8	
F9	
FA	
FB	

Adresa	Sadržaj
FC	
FD	
FE	
FF	
100	
101	
102	
103	
104	
105	
106	
107	

Upišite u tablicu desno stanje svih memorijskih lokacija od F0 do 107 i **strelicom** označite položaj SP nakon izvođenja gornjeg programa. Početno su sve prikazane memorijske lokacije u 0.

1 b) (*2,5 boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **STOREB R1, (R2+3ABC)**:

Razina dohvata:

Rastući brid CLOCK-a:

_____

Padajući brid CLOCK-a:

_____

_____

_____

_____

Razina izvođenja:

Rastući brid CLOCK-a:

_____

_____

_____

Padajući brid CLOCK-a:

_____

_____

1 c) (*1,5 boda*) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: _____, _____, _____, _____, _____ i _____.

1 d) (*0,5 boda*) Za procesor FRISC, kod naredbe LOAD dolazi do pojave _____ hazarda, a kod naredbe PUSH dolazi do pojave _____ hazarda.

1 e) *(1,5 bod)* Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0110 - 1100. Nakon operacije će biti: prijenos = ____ , posudba = _____ , preljev = _____ , ništica = _____ , predznak = _____ . **Potrebno je napisati postupak rješenja.**

1 f) (*3 boda*) Odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**):

```
DVA     `EQU 2          Trajanje___   Izvodi se___ puta
        `ORG 0          Trajanje___   Izvodi se___ puta
        MOVE   4,  R0   Trajanje___   Izvodi se___ puta
POC     SUB    R0, DVA, R0  Trajanje___   Izvodi se___ puta
        JR_NE  POC      Trajanje___   Izvodi se___ puta
        STORE  R0, (1000)  Trajanje___   Izvodi se___ puta
        HALT            Trajanje___   Izvodi se___ puta
```

Kraj **svake** naredbe napišite njeno **trajanje** u ciklusima, a zatim **koliko** puta se naredba izvodi. Izvođenje cijelog programa ukupno traje _____ ciklusa.

2. (*4,5 boda*) U memoriji se nalazi blok 8-bitnih brojeva u zapisu s bitom za predznak. Adresa početka bloka je 2000_{16}, a broj 8-bitnih podataka u bloku zapisan je na adresi BROJPOD.

Napišite program koji 8-bitne brojeve u zapisu s bitom za predznak pretvara u 24-bitne brojeve u zapisu 2'k. Također je za svaki broj potrebno odrediti je li paran ili neparan. Pretvorene brojeve (u redoslijedu *little-endian*) treba spremiti u novi blok memorije, počevši od adrese 3000_{16}, a nakon svakog zapisanog 24-bitnog broja, u sljedećih 8 bita potrebno je zapisati vrijednost 1 ako je broj neparan, a 0 ako je broj paran.

3. (7 *bodova*) U memoriji se nalazi blok četveroznamenkastih dekadskih brojeva zapisanih u 16 bita kao pakirani BCD. Blok podataka nalazi se na adresi 1000_{16}, a završava podatkom 0_{16} (ovaj se podatak ne smatra brojem u bloku).

Napišite glavni program koji za svaki broj poziva potprogram PARNOST, a za neparne brojeve poziva potprogram PRETVORI. Pretvorene brojeve glavni program sprema u memoriju od adrese 4000_{16}.

Napišite potprogram PARNOST koji provjerava je li četveroznamenkasti dekadski broj paran. Dekadski broj je zapisan u 16 bita kao pakirani BCD (svaka znamenka dekadskog broja zauzima 4 bita). Glavni program preko stoga šalje potprogramu broj koji je potrebno provjeriti. Potprogram preko registra R0 vraća vrijednost 0 ako je broj paran, a 1 ako je broj neparan.

Napišite potprogram PRETVORI koji zapis četveroznamenkastog dekadskog broja, zapisanog u 16 bita kao pakirani BCD pretvara u 32 bita kao nepakirani BCD (svaka znamenka dekadskog broja zauzima 8 bita)**.** Potprogram preuzima parametar preko memorijske lokacije zadane labelom PAKBCD, a rezultat vraća preko memorijske lokacije zadane labelom NEPAKBCD.

Primjer zapisa:

Dekadski broj	Pakirani BCD (16-bita)	Nepakirani BCD (32-bita)
5432_{10}	0101 0100 0011 0010 5 4 3 2	0000 0101 0000 0100 0000 0011 0000 0010 5 4 3 2

4. (6,5 *bodova*) U računalnom sustavu nalazi se procesor FRISC i četiri vanjske jedinice: dvije uvjetne UVJ1 i UVJ2, i dvije prekidne PVJ3 (spojena na INT0) i PVJ4 (spojena na INT3). Adrese vanjskih jedinica odaberite sami.

Napišite program koji preuzima 32-bitne podatke u zapisu 2'k s međusobno nezavisnih jedinica UVJ1 i UVJ2. Program na memorijskoj lokaciji BROJAC čuva broj ukupno primljenih podataka (s obje vanjske jedinice zajedno), a također na memorijskoj lokaciji ZBROJ čuva sumu svih primljenih podataka.

Na svaki zahtjev prekidne vanjske jedinice PVJ3, potrebno je provjeriti je li zbroj pozitivan ili negativan. Ako je zbroj pozitivan, treba ga poslati na PVJ3, a ako je zbroj negativan, na PVJ3 treba poslati podatak $1234FFFF_{16}$.

Na svaki zahtjev prekidne vanjske jedinice PVJ4, potrebno joj je poslati broj svih primljenih podataka.

Glavni program izvodi se beskonačno.

Međuispit iz Arhitekture računala 1 **20. travnja 2012.** **Grupa na predavanjima: _____**

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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Međuispit traje 120 minuta.

1 a) (*3 boda*) FRISC izvodi sljedeći program:

```
        `ORG 0
        MOVE  100, SP
        MOVE  0FFFFABCD, R0
        PUSH   R0
        CALL    POTP
        ADD    SP, 8, SP
        CMP    R7,100
        HALT_NZ
        PUSH   R0
        HALT
POTP    MOVE   14, R0
        PUSH   R0
        RET
```

Upišite u tablicu desno stanje svih memorijskih lokacija od F0 do 107 i strelicom označite položaj SP nakon izvođenja gornjeg programa. Početno su sve prikazane memorijske lokacije u 0.

Adresa	Sadržaj
F0	00
F1	00
F2	00
F3	00
F4	14
F5	00
F6	00
F7	00
---> F8	10
F9	?0
FA	00
FB	00

Adresa	Sadržaj
FC	CD
FD	AB
FE	FF
FF	FF
100	00
101	00
102	00
103	00
104	00
105	00
106	00
107	00

1 b) (*2,5 boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **STOREB R1, (R2+3ABC)**:

Razina dohvata:

Rastući brid CLOCK-a:

____**PC -> AR**_____

Padajući brid CLOCK-a:

____**PC +4 -> PC**_____

____**(AR) -> IR, dekodiranje** _____

____**ext 3ABC i R2 -> ALU** _____

_ **onemogući dohvat u sljedećem ciklusu** _

Razina izvođenja:

Rastući brid CLOCK-a:

___**ALU: izvodi zbrajanje** _____

___**ALU -> AR** _____

___**R1 -> DR**_____

Padajući brid CLOCK-a:

__**DR -> (AR)**_____

__ **omogući dohvat u sljedećem ciklusu** ___

1 c) (*1,5 boda*) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: _**ADR**_, _**DATA**__, _**READ**__, ____**WRITE**____, _____**WAIT**__ i ___**SIZE**___.

1 d) (*0,5 boda*) Za procesor FRISC, kod naredbe LOAD dolazi do pojave ___**strukturnog**___ hazarda, a kod naredbe PUSH dolazi do pojave ___**strukturnog**___ hazarda.

1 e) *(1,5 bod)* Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0110 - 1100. Nakon operacije će biti: prijenos = _**0**_ , posudba = _**1**_ , preljev = _**1**_ , ništica = _**0**_ , predznak = _**1**_ . **Potrebno je napisati postupak rješenja.**

1 f) (*3 boda*) Odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**):

```
DVA     `EQU 2
        `ORG 0
        MOVE   4,  R0
POC     SUB    R0, DVA, R0
        JR_NE  POC
        STORE  R0, (1000)
        HALT
```

Trajanje_**0**_ Izvodi se_**0**_ puta
Trajanje_**0**_ Izvodi se_**0**_ puta
Trajanje_**1**_ Izvodi se_**1**_ puta
Trajanje_**1**_ Izvodi se_**2**_ puta
Trajanje_**2**_ Izvodi se_**2**_ puta
Trajanje_**2**_ Izvodi se_**1**_ puta
Trajanje_**2**_ Izvodi se_**1**_ puta

Kraj **svake** naredbe napišite njeno **trajanje** u ciklusima, a zatim **koliko** puta se naredba izvodi. Izvođenje cijelog programa ukupno traje ___**11**_____ ciklusa.

2. (*4,5 boda*) U memoriji se nalazi blok 8-bitnih brojeva u zapisu s bitom za predznak. Adresa početka bloka je 2000_{16}, a broj 8-bitnih podataka zapisan je na adresi BROJPOD.

Napišite program koji 8-bitne brojeve u zapisu s bitom za predznak pretvara u 24-bitne brojeve u zapisu 2'k. Također je za svaki broj potrebno odrediti je li paran ili neparan. Pretvorene brojeve (u redoslijedu *little-endian*) treba spremiti u novi blok memorije, počevši od adrese 3000_{16}, a nakon svakog zapisanog 24-bitnog broja, u sljedećih 8 bita potrebno je zapisati vrijednost 1 ako je broj neparan, a 0 ako je broj paran.

```
      `ORG 0

      MOVE 2000, R0         ; R0 - adresa izvorišnog bloka
      MOVE 3000, R1         ; R1 - adresa odredišnog bloka
      LOAD R6, (BROJPOD)    ; R6 - broj podataka

POC   LOADB R2, (R0)        ; učitavanje 8-bitnog podatka => R2
      ADD R0, 1, R0         ; povećanje lokacije za novi podatak

      MOVE 0, R5            ; R5 – parnost: 0 (paran) ili 1 (neparan)
      ROTR R2, 1, R3        ; provjera parnosti; ili AND 1 + JR_NZ;
      JR_NC DALJE
NEP   MOVE 1, R5            ; 1 za najviši bajt (neparan)
                            ; inače ostaje 0 (za parne brojeve)
DALJE
      AND R2, 80, R3        ; provjera predznaka, ILI SHL/ROTL 25; _NC
      JR_Z SPREMI
      AND R2, 7F, R3        ; brisanje predznaka

NEG   XOR R2, -1, R2        ; negativni brojevi -> 2'k
      ADD R2, 1, R2

SPREMI
      STOREB R2, (R1)       ; spremanje donja tri bajta, little-endian
      ROTR R2, 8, R2        ; ILI petlja 3 puta, ili STOREH+STOREB
      STOREB R2, (R1+1)     ; ILI dodavanje parnosti s OR/ADD direktno
      ROTR R2, 8, R2        ; (na 24-31 bit), pa onda STORE + ADD 4
      STOREB R2, (R1+2)     ; (paziti da se ne zapiše 1 s MOVE u 24.bit)

      STOREB R5, (R1+3)     ; spremanje parnosti
      ADD R1, 4, R1         ; povećavanje odredišne adrese za 4

      SUB R6, 1, R6         ; smanjivanje brojača
      JP_NZ POC             ; petlja na početak

      HALT                  ; zaustavljanje procesora

BROJPOD    DW    31245234

      `ORG 2000
BLOK  DW    .....
```

3. (7 *bodova*) U memoriji se nalazi blok četveroznamenkastih dekadskih brojeva zapisanih u 16 bita kao pakirani BCD (svaka znamenka dekadskog broja zauzima 4 bita). Blok podataka nalazi se na adresi 1000_{16}, a završava podatkom 0_{16} (ovaj se podatak ne smatra brojem u bloku).

Napišite glavni program koji za svaki broj poziva potprogram PARNOST, a za neparne brojeve poziva potprogram PRETVORI. Pretvorene brojeve glavni program sprema u memoriju od adrese 4000_{16}.

Napišite potprogram PARNOST koji provjerava je li četveroznamenkasti dekadski broj paran. Dekadski broj je zapisan u 16 bita kao pakirani BCD (svaka znamenka dekadskog broja zauzima 4 bita). Glavni program preko stoga šalje potprogramu broj koji je potrebno provjeriti. Potprogram preko registra R0 vraća vrijednost 0 ako je broj paran, a 1 ako je broj neparan.

Napišite potprogram PRETVORI koji zapis četveroznamenkastog dekadskog broja, zapisanog u 16 bita kao pakirani BCD pretvara u 32 bita kao nepakirani BCD (svaka znamenka dekadskog broja zauzima 8 bita)**.** Potprogram preuzima parametar preko memorijske lokacije zadane labelom PAKBCD, a rezultat vraća preko memorijske lokacije zadane labelom NEPAKBCD.

Primjer zapisa:

Dekadski broj	Pakirani BCD (16-bita)	Nepakirani BCD (32-bita)
5432_{10}	0101 0100 0011 0010 5 4 3 2	0000 0101 0000 0100 0000 0011 0000 0010 5 4 3 2

```
      'ORG 0
      MOVE 10000, SP        ; inicijalizacija pokazivača stoga
      MOVE 1000, R2         ; R2 – adresa izvorišnog bloka
      MOVE 4000, R3         ; R3 – adresa odredišnog bloka
PONOVI
      LOADH R1, (R2)        ; učitavanje podatka
      CMP R1, 0             ; ako je 0 – nema više podataka
      HALT_EQ               ; -> kraj programa
      ADD R2, 2, R2         ; povećavanje lokacije za čitanje podatka

      PUSH R1               ; stavljanje podatka na stog
      CALL PARNOST          ; poziv potprograma za provjeru
      ADD SP, 4, SP         ; brisanje parametra sa stoga
      CMP R0, 0             ; 0 = paran; povratak na početak
      JR_EQ PONOVI          ; 1 = neparan;
VECIJ STORE R1, (PAKBCD)    ; spremanje parametra u memoriju (može H)
      CALL PRETVORI         ; poziv potprograma za pretvorbu
      LOAD R0, (NEPAKBCD)   ; učitavanje rezultata iz memorije
      STORE R0, (R3)        ; spremanje NEPAKBCD u blok
      ADD R3, 4, R3         ; povećavanje lokacije za spremanje
      JR PONOVI             ; povratak na učitavanje novog broja
KRAJ  HALT                  ; zaustavljanje procesora
-------------
PARNOST
      LOAD R0, (SP+4)       ; dohvaćanje podatka sa stoga
      AND R0, 1, R0         ; izravna provjera i definiranje parnosti
                            ; može i odvojeno provjera, pa upis u R0
      RET                   ; povratak iz potprograma
```

```
-------------
PRETVORI
     PUSH R0              ; spremanje konteksta
     PUSH R1
     PUSH R2
     PUSH R3
     PUSH R4

     LOAD R0, (PAKBCD)    ; učitavanje parametra iz memorije

     AND R0,    0F, R1    ; 1. znamenka, čišćenje
     AND R0,   0F0, R2    ; 2. znamenka, čišćenje
     AND R0,  0F00, R3    ; 3. znamenka, čišćenje
     AND R0, 0F000, R4    ; 4. znamenka, čišćenje

     SHL R2, 4, R2        ; 2. znamenka, pomicanje
     SHL R3, 8, R3        ; 3. znamenka, pomicanje
     SHL R4, %D 12, R4    ; 4. znamenka, pomicanje

     OR R1, R2, R1        ; dodavanje u cjelinu
     OR R1, R3, R1
     OR R1, R4, R1

     STORE R1, (NEPAKBCD) ; spremanje rezultata u memoriju

     POP R4
     POP R3
     POP R2               ; vraćanje konteksta
     POP R1
     POP R0
     RET                  ; povratak iz potprograma
```

4. (6,5 *bodova*) U računalnom sustavu nalazi se procesor FRISC i četiri vanjske jedinice: dvije uvjetne UVJ1 i UVJ2, i dvije prekidne PVJ3 (spojena na INT0) i PVJ4 (spojena na INT3). Adrese vanjskih jedinica odaberite sami.

Napišite program koji preuzima 32-bitne podatke u zapisu 2'k s međusobno nezavisnih jedinica UVJ1 i UVJ2. Program na memorijskoj lokaciji BROJAC čuva broj ukupno primljenih podataka (s obje vanjske jedinice zajedno), a također na memorijskoj lokaciji ZBROJ čuva sumu svih primljenih podataka.

Na svaki zahtjev prekidne vanjske jedinice PVJ3, potrebno je provjeriti je li zbroj pozitivan ili negativan. Ako je zbroj pozitivan, treba ga poslati na PVJ3, a ako je zbroj negativan, na PVJ3 treba poslati podatak $1234FFFF_{16}$.

Na svaki zahtjev prekidne vanjske jedinice PVJ4, potrebno joj je poslati broj svih primljenih podataka.

Glavni program izvodi se beskonačno.

```
VJ1_PRIMI  `EQU  0FFFF1000       ; adrese vanjskih jedinica
VJ1_STANJE `EQU  0FFFF1004

VJ2_PRIMI  `EQU  0FFFF2000
VJ2_STANJE `EQU  0FFFF2004

PVJ3_POD   `EQU  0FFFF3000
PVJ3_IACK  `EQU  0FFFF3004
PVJ3_IEND  `EQU  0FFFF3008
PVJ3_STOP  `EQU  0FFFF300C

PVJ4_POD   `EQU  0FFFF4000
PVJ4_IEND  `EQU  0FFFF4008
PVJ4_STOP  `EQU  0FFFF400C

      `ORG  0
      MOVE  10000, SP
      JP    GLAVNI

      `ORG  8
      DW    1000                ; prekidni vektor
--------
      `ORG  0C                  ; NMI
      PUSH R0                   ; čuvanje konteksta; ne treba SR (samo LOAD)
      LOAD R0, (BROJAC)         ; učitavanje broja podataka
      STORE R0, (PVJ4_POD)      ; slanje broja podataka
      STORE R0, (PVJ4_IEND)     ; dojava kraja posluživanja
      POP R0                    ; vraćanje konteksta
      RETN                      ; povratak iz nemaskirajućeg prekida
--------
GLAVNI
      MOVE  %B 10010000, SR     ; dozvoli prekide INT0

PRVA  LOAD  R0, (VJ1_STANJE)    ; ispitivanje spremnosti VJ1
      OR    R0, R0, R0
      JP_Z  DRUGA               ; ako nije spremna, prozivanje druge VJ2

      LOAD  R0, (VJ1_PRIMI)     ; primanje podatka s VJ1
      LOAD  R1, (BROJAC)        ; učitavanje trenutne vrijednosti brojača
      ADD   R1, 1, R1           ; povećavanje brojača
      STORE R1, (BROJAC)        ; spremanje
      LOAD  R1, (ZBROJ)         ; učitavanje trenutne vrijednosti zbroja
      ADD   R1, R0, R1          ; pribrajanje podatka zbroju
      STORE R1, (ZBROJ)         ; spremanje
      STORE R0, (VJ1_STANJE)    ; brisanje spremnosti

DRUGA LOAD  R0, (VJ2_STANJE)    ; ispitivanje spremnosti VJ2
      OR    R0, R0, R0
      JP_Z  PRVA                ; ako nije spremna, prozivanje prve VJ1
```

```
      LOAD  R0, (VJ2_PRIMI)       ; primanje podatka s VJ2
      LOAD  R1, (BROJAC)          ; učitavanje trenutne vrijednosti brojača
      ADD   R1, 1, R1             ; povećavanje brojača
      STORE R1, (BROJAC)          ; spremanje
      LOAD R1, (ZBROJ)            ; učitavanje trenutne vrijednosti zbroja
      ADD   R1, R0, R1            ; pribrajanje podatka zbroju
      STORE R1, (ZBROJ)           ; spremanje
      STORE R0, (VJ2_STANJE)
      JP    PRVA                  ; natrag na prozivanje VJ1

BROJAC DW 0
ZBROJ DW 0
PODATAK DW 1234FFFF

---------
; PREKIDNI POTPROGRAM
      `ORG  1000
      PUSH  R0                    ; spremanje konteksta
      MOVE  SR, R0
      PUSH  R0

      STORE R0, (PVJ3_IACK)       ; prihvat prekida
      LOAD  R0, (ZBROJ)           ; učitavanje zbroja
      OR R0, R0, R0               ; postavljanje zastavica, moze i ROTL
      JR_P  SPREMI                ; ili SHL, pa gledanje carryja

NEG   LOAD R0, (PODATAK)          ; učitavanje podatka; prevelik za MOVE
SPREMI STORE R0, (PVJ3_POD)       ; slanje podatka
      STORE R0, (PVJ3_IEND)       ; dojava kraja posluživanja

      POP   R0                    ; obnova konteksta
      MOVE  R0, SR
      POP   R0
      RETI                        ; povratak iz maskirajućeg prekida
---------
```

2. međuispit iz Arhitekture računala 1 18.5.2006.

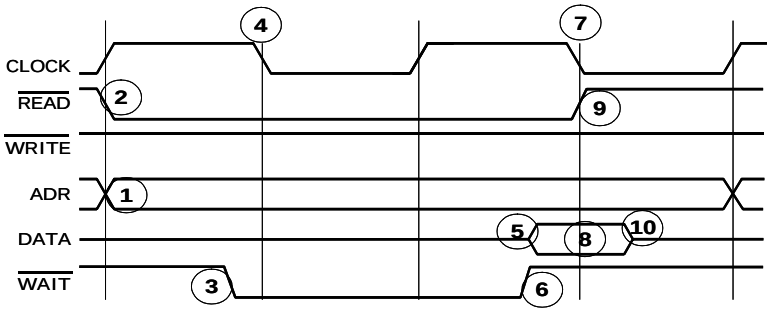
Prezime i ime (štampano):_____ MBR: _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Potpis:_____

Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa.

PREPORUČUJEMO
da test popunjavate običnom olovkom kako bi u slučaju greške mogli napraviti ispravke



Zadatak 1. (6 bodova)

1.a) (3 boda) Na slici je za FRISC prikazan sabirnički protokol čitanja iz memorije s jedinim stanjem čekanja. Na slici je označeno 10 koraka u komunikaciji. Ispod su ponuđeni koraci među kojima odaberite 10 onih koji odgovaraju koracima na slici. Uz odabrane korake na prazne crte upišite brojeve 1 do 10.

_____FRISC ispituje WAIT i postavlja podatak na podatkovnu sabirnicu

_____Nakon što je postavljen podatak na podatkovnu sabirnicu, procesor deaktivira WAIT

_____FRISC aktivira WAIT jer čita iz spore memorije

__9___FRISC deaktivira priključak READ

__5___Memorija dovršava traženu operaciju čitanja i postavlja podatak na podatkovnu sabirnicu

__6___Memorija deaktivira WAIT

__8___FRISC preuzima podatak s podatkovne sabirnice u DR

__7___FRISC ispituje WAIT i prepoznaje da je memorija obavila traženu operaciju

__2___FRISC aktivira READ čime naznačuje da želi čitati podatak

__4___FRISC ispituje WAIT i dodaje stanje čekanja

_____Memorija postavlja na adresnu sabirnicu adresu podatka koji će se čitati

___3__Memorija je spora pa zato aktivira WAIT kako bi je FRISC pričekao

_____FRISC postavlja podatak iz DR na podatkovnu sabirnicu

___1__FRISC postavlja adresu iz AR na adresnu sabirnicu

_____Memorija aktivira READ na početku operacije čitanja

_____Memorija deaktivira READ nakon što je WAIT postao neaktivan

_____FRISC nakon čitanja podatka postavlja sabirnicu podataka u visoku impedanciju

__10__Nakon deaktiviranja READ, memorija postavlja sabirnicu podataka u visoku impedanciju

Nastavak zadataka je na poleđini >>>

1.b) (3 boda) Na prazne crte upišite korake (npr. PC+4 → PC) koje FRISC obavlja prilikom izvođenja aritmetičko-logičke naredbe `ADD R1,R2,R3`:

Razina dohvata:

Rastući brid CLOCK-a:

_____PC → AR_____

Padajući brid CLOCK-a:

_____PC+4 → PC_____

_____(AR) → IR, dekodiranje_____

_____R1 i R2 → ALU_____

Razina izvođenja:

Rastući brid CLOCK-a:

____ALU: izvodi zbrajanje_____

Padajući brid CLOCK-a:

____ALU → R3_____

____postavljanje zastavica u SR-u_____

Zadatak 2. (7 bodova)

Na FRISC su spojene dvije uvjetne jedinice VJ1 i VJ2 (na adresama FFFF 1000 i FFFF 2000), PIO (adresa FFFF 3000) i CT (adresa FFFF 4000).

Napišite program koji beskonačno prima 32-bitne podatke sa VJ1 i VJ2, i redom dolaska ih šalje na PIO (PIO radi kao bezuvjetna jedinica). Zbog jednostavnosti, pretpostavite da je opseg svih podataka sa VJ1 i VJ2 takav da stane u jedan bajt i da se može poslati na PIO bez gubitka informacija.

VJ2 osim uobičajenih dvaju adresa zauzima i dodatnu treću adresu. Slanje bilo kojeg podatka na treću adresu će resetirati VJ2.

CT generira prekid na INT0 svake milisekunde (na CNT je spojen CLOCK frekvencije 20 MHz). U prekidnom potprogramu treba provjeriti koliko je podataka poslala VJ2 u protekloj milisekundi. Ako VJ2 nije poslala niti jedan podatak, onda pretpostavljamo da se "zaglavila" i treba je resetirati.

```
;;;;;; definicije labela za adresiranje vanjskih jedinica

VJ1_DATA   `EQU   FFFF1000
VJ1_STATUS `EQU   FFFF1004

VJ2_DATA   `EQU   FFFF2000
VJ2_STATUS `EQU   FFFF2004
VJ2_RESET  `EQU   FFFF2008

PIO_CR     `EQU   FFFF3000
PIO_DATA   `EQU   FFFF3004

CT_LR      `EQU   FFFF4000
CT_CR      `EQU   FFFF4004
CT_IACK    `EQU   FFFF4008
CT_IEND    `END   FFFF400C


       `ORG  0
       MOVE  10000, SP    ; inicijalizacija stoga i skok u glavni
       JP    GLAVNI
```

```
       `ORG  8
       DW    1000          ; prekidni vektor


;;;;;; GLAVNI PROGRAM

GLAVNI  ;;;;;; init PIO

        MOVE  %B 100, R0      ; način postavljanja bitova
        STORE R0, (PIO_CR)

        ;;;;;; init CT

        MOVE  %D 20000, R0    ; konstanta za dijeljenje 20 MHZ na 1 kHZ
        STORE R0, (CT_LR)
        MOVE  %B 11, R0       ; brojenje s prekidom
        STORE R0, (CT_CR)

        MOVE  0, R0           ; brojač prenesenih podataka na VJ1
        STORE R0, (BROJAC)

        MOVE %B 10010000, SR  ; dozvoli prekide


        ;;;;;; Prozivanje VJ1 i VJ2

POLL    LOAD  R0, (VJ1_STATUS)    ; ispitivanje VJ1
        AND   R0, 1, R0
        CALL_NZ  PVJ1

        LOAD  R0, (VJ2_STATUS)    ; ispitivanje VJ2
        AND   R0, 1, R0
        CALL_NZ  PVJ2

        JR POLL ; ponavljaj zauvijek


;;;; Posluži VJ1

PVJ1    LOAD  R0, (VJ1_DATA)    ; prenesi podatak sa VJ1 -> PIO
        STORE R0, (PIO_DATA)

        STORE R0, (VJ1_STATUS)  ; briši status VJ1
        RET


;;;; Posluži VJ2

PVJ1    LOAD  R0, (VJ2_DATA)    ; prenesi podatak sa VJ2 -> PIO
        STORE R0, (PIO_DATA)

        STORE R0, (VJ2_STATUS)  ; briši status VJ2

        LOAD  R0, (BROJAC)      ; povečaj brojač prenesenih podataka za VJ2
        ADD   R0, 1, R0
        STORE R0, (BROJAC)
        RET


;;;;; Lokacija za brojač

BROJAC  `DW   0


;;;;;; Prekidni potprogram na adresi 1000

        `ORG 1000
```

```
        PUSH  R0        ; spremi kontekst (R0 i SR)
        MOVE  SR, R0
        PUSH  R0

        STORE R0, (CT_IACK)  ; dojavi prihvat prekida CT-u

        LOAD  R0, (BROJAC)    ; provjeri brojač
        CMP   R0, 0
        JR_NE VAN            ; ako je bilo prijenosa sa VJ2 => onda izađi iz p.p.

        STORE R0, (VJ2_RESET)  ; ako nije bilo prijenosa sa VJ2 => resetiraj VJ2

VAN     MOVE  0, R0
        STORE R0, (BROJAC)     ; brisanje brojača za sljedeću milisekundu

        STORE R0, (CT_IEND)    ; dojava kraja posluživanja CT-u

        POP   R0        ; obnovi kontekst
        MOVE  R0, SR
        POP   R0
        RETI
```

Zadatak 3. (7 bodova)

Na FRISC su spojena bezuvjetna VJ1 (na adresi FFFF FF00) i FRISC-DMA (na adresi FFFF 0000). Prvo treba DMA-prijenosom prenijeti 1000_{16} podataka iz VJ1 u memoriju. Adresa memorijskog bloka u kojeg se pune podatci je 3000_{16}. Prijenos treba obaviti krađom ciklusa i bez generiranja prekida.

Podatci koji se primaju od VJ1 su 32-bitni brojevi u formatu 2'k i sigurno su u opsegu od -100_{10} do $+100_{10}$. Nakon prijenosa treba na kraj bloka podataka postaviti oznaku 80000000_{16} (iza zadnjeg podatka). Nakon toga treba za taj blok pozvati potprogram PROMIJENI. Na kraju treba zaustaviti glavni program.

Potprogram PROMIJENI prima preko stoga jedan parametar (uklanja ga pozivatelj): početnu adresu bloka. Potprogram nema povratne vrijednosti i mora čuvati vrijednosti registara. Zadaća potprograma je da sve negativne podatke u bloku zamijeni s njihovim pozitivnim vrijednostima. Veličina bloka nije poznata unaprijed, ali se zna da je blok zaključen podatkom 80000000_{16}. Na primjer, ako su u bloku podatci: 12, -3, 1A1, 2, -5, 80000000, onda taj blok nakon potprograma mora biti: 12, 3, 1A1, 2, 5, 80000000.

```
;;;;;; definicije labela za adresiranje vanjskih jedinica

VJ1         `EQU   FFFFFF00

DMA_SRC     `EQU   FFFF4000
DMA_DEST    `EQU   FFFF4004
DMA_SIZE    `EQU   FFFF4008
DMA_CR      `END   FFFF400C
DMA_GO      `EQU   FFFF4010
DMA_CLR     `EQU   FFFF4014

;;;;;; GLAVNI PROGRAM

        `ORG  0

        MOVE  10000, SP    ; inicijalizacija stoga

GLAVNI  ;;;;;; init DMA

        MOVE  0FFFFFF00, R0   ; adresa izvora (VJ1)
        STORE R0, (DMA_SRC)

        MOVE  3000, R0        ; adresa odredišta (mem. blok)
        STORE R0, (DMA_DEST)

        MOVE  1000, R0     ; broj podataka
```

```
       STORE R0, (DMA_SIZE)

       MOVE  %B 0110, R0     ; upravljačka riječ: VJ->mem. krađa ciklusa, bez prekida
       STORE R0, (DMA_CR)

       STORE R0, (DMA_GO)    ; pokreni DMA-prijenos


CEKAJ  ;;;;;; Čekaj da završi DMA-prijenos

       LOAD  R0, (DMA_CR)
       AND   R0, 1, R0
       JR_Z  CEKAJ

;;;;;;;  Dio programa koji se izvodi nakon DMA-prijenosa

       LOAD  R0, (OZNAKA)     ; označi kraj bloka
       STORE R0, (7000)       ; (početna adresa 3000) + (1000 podataka) * (4 bajta)

       MOVE  3000, R0         ; parametar na stog: adresa bloka
       PUSH  R0

       CALL  PROMIJENI        ; obradi blok pomoću potprograma PROMIJENI

       ADD   SP, 4, SP        ; ukloni parametar sa stoga

       HALT


;;;;;;;;; Potprogram PROMIJENI
;
;   R1 = pokazivač na podatak koji se čita
;   R2 = oznaka kraja bloka
;   R0 služi za učitavanje pojedinog podatka iz bloka
;
;
;   Stog (adresa i sadrzaj):
;
;     SP+0:       R2
;     SP+4:       R1
;     SP+8:       R0
;     SP+C:       povratna adresa
;     SP+10:      parametar: adresa bloka
;

PROMIJENI
       PUSH  R0      ; spremi registre
       PUSH  R1
       PUSH  R2

       LOAD  R1, (SP+10)  ; dohvati adresu bloka
       LOAD  R2, (OZNAKA) ; R2=oznaka kraja

LOOP   LOAD  R0, (R1)     ; učitaj podatak iz bloka

       CMP   R0, R2       ; provjeri je li kraj bloka
       JR_EQ KRAJ

       CMP   R0, 0        ; provjeri je li pozitivan ili negativan
       JR_SGE POZIT

NEGAT  XOR   R0, -1, R0   ; promijeni mu predznak
       ADD   R0, 1, R0

       STORE R0, (R1)     ; kopiraj pozitivan preko negativnog

POZIT  ADD   R1, 4, R1    ; pomakni se na sljedeći broj
       JR    LOOP         ; ponavljaj petlju

KRAJ   POP   R2    ; obnovi registre
```

```
        POP   R1
        POP   R0

        RET


OZNAKA DW 80000000
```

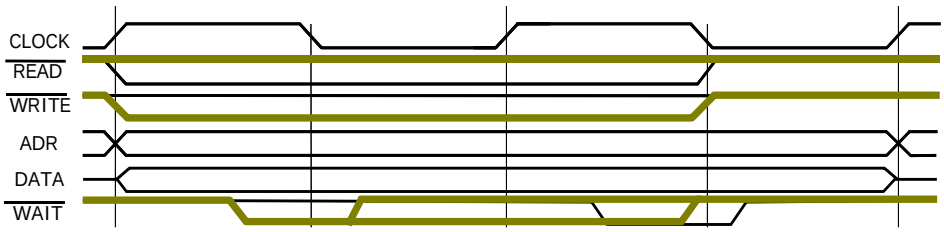
2. Međuispit iz Arhitekture računala 1 17.5.2007.

Prezime i ime (štampano):_____ MBR: _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____

Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa.

PREPORUČUJEMO da test popunjavate običnom olovkom kako bi u slučaju greške mogli napraviti ispravke



Zadatak 1. (2 boda) Gornja slika prikazuje sabirnički protokol pisanja u memoriju s jednim stanjem čekanja za FRISC

a) Tri signala su prikazana pogrešno. Olovkom na slici nacrtajte kako bi oni trebali izgledati.

b) Na crte pokraj imena pojedinih signala upišite koja komponenta računalnog sustava upravlja dotičnim signalom na sabirnici (za protokol pisanja u memoriju): READ___procesor (ili FRISC)___ WRITE___procesor___, ADR___procesor___, DATA___procesor_____, WAIT___memorija___

Zadatak 2. (2 boda) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe `LOAD R1,(50)`:

Razina dohvata:

Rastući brid CLOCK-a:

__PC → AR_____

Padajući brid CLOCK-a:

__PC + 4 → PC_____

__(AR) → IR, dekodiranje__ (bitne su zagrade oko AR)

__ext 50 (ili predznačno proširi 50)

__onemogući dohvat u sljedećem ciklusu (ili sljedeće naredbe)__

Razina izvođenja:

Rastući brid CLOCK-a:

__ext 50 → AR_____

Padajući brid CLOCK-a:

__(AR) → DR_____ (bitne su zagrade oko AR)

__DR → R1_____

__omogući dohvat u sljedećem ciklusu__

Zadatak 3. (2 boda)

a) Sklop FRISC-DMA može raditi u dva načina DMA-prijenosa: _krađa ciklusa_____ i __zaustavljanje procesora_____. Način DMA-prijenosa koji kombinira dva prethodna (FRISC-DMA ga inače nema) zove se _____blokovski prijenos_____.

b) FRISC dobiva od sklopa FRISC-DMA zahtijev za upravljanje nad sabirnicom preko priključka ____BREQ_____, a potvrdu da je predao sabirnicu FRISC dojavljuje priključkom ____BACK_____.

c) Kada FRISC-DMA radi u načinu rada ____zaustavljanja procesora_____, onda nije potrebno ispitivati kraj DMA-prijenosa. U protivnom se kraj prijenosa mora ispitati, a to se obično radi pomoću _____prekida_____i pomoću ___uvjetno (ili ispitivanja stanja DMA-sklopa)___.

Preostali zadatci su na poleđini >>>

Zadatak 4. (7 bodova)

Na FRISC su spojene uvjetna vanjska jedinica VJ1 (adresa $FFFF1000_{16}$), bezuvjetna vanjska jedinica VJ2 (adresa $FFFF2000_{16}$) i CT (adresa $FFFF3000_{16}$, spojen na INT0). FRISC radi na 10 MHz.

U glavnom programu se 32-bitni podatci primaju s uvjetne VJ1 i spremaju u memoriju u blok koji počinje na adresi 1000_{16}. U memorijskoj lokaciji BROJAC glavni program treba prebrajati koliko je podataka primio sa VJ1. Kad se primi 3000_{16} podataka treba zaustaviti program.

CT treba generirati prekid svake 2 milisekunde. U svakom pozivu prekidnog potprograma treba poslati na bezuvjetnu VJ2 trenutačnu vrijednost lokacije BROJAC.

```
VJ1_PRIMI     'EQU 0FFFF1000                ; adrese vanjskih jedinica
VJ1_ISPITAJ   'EQU 0FFFF1004
VJ1_BRISI     'EQU 0FFFF1004
VJ2           'EQU 0FFFF2000
CTLR          'EQU 0FFFF3000
CTCR          'EQU 0FFFF3004
CTIACK        'EQU 0FFFF3008
CTIEND        'EQU 0FFFF300C

              'ORG 0                        ; inicijalizacija SP i skok u glavni
              MOVE 10000, R7
              JP GLAVNI

              'ORG 8                        ; prekidni vektor
              'DW 500

GLAVNI        MOVE 1000,R0                  ; inicijalizacije pokazivača i brojača
              MOVE 3000,R1

              MOVE %D 20000,R2              ; vremenska konstanta za 2 ms
              STORE R2,(CTLR)
              MOVE %B11,R2                  ; upravljačka riječ za CT
              STORE R2,(CTCR)

              MOVE %B10010000, SR           ; dozvoli prekide

CEKAJ         LOAD R2,(VJ1_ISPITAJ)         ; čekaj spremnost VJ1
              OR R2, R2, R2
              JR_Z CEKAJ

              LOAD R2,(VJ1_PRIMI)           ; primi podatak
              STORE R2,(VJ1_BRISI)

              STORE R2,(R0)                 ; spremi podatak u memoriju

              LOAD R3, (BROJAC)             ; povećanje brojača primljenih podataka
              ADD R3, 1, R3
              STORE R3, (BROJAC)

              ADD R0, 4, R0                 ; povećaj pokazivač i smanji brojač
              SUB R1, 1, R1
              JR_NZ CEKAJ                   ; kraj petlje i zaustavljanje
              HALT

BROJAC        'DW 0                         ; lokacija za brojač primljenih podataka


              'ORG 500                      ; adresa prekidnog potrograma

              PUSH R0                       ; pohrana konteksta

              STORE R0,(CTIACK)             ; dojava prihvata prekida

              LOAD R0, (BROJAC)             ; dohvat brojača i slanje na VJ2
              STORE R0,(VJ2)

              STORE R0,(CTIEND)             ; dojava kraja prekida (može i ispred RETI)

              POP R0                        ; obnova konteksta s povratkom
              RETI
```

Zadatak 5. (7 bodova)

Na FRISC su spojeni PIO (adresa $FFFF1000_{16}$, spojen na INT0) i uvjetna vanjska jedinica VJ1 (adresa $FFFF2000_{16}$). Na priključke PIOD0 i PIOD1 spojena su dva senzora tlaka (PIOD0 za senzor 1 i PIOD1 za senzor 2). Senzori dojavljuju prekoračenje nazivnog tlaka slanjem pozitivnog impulsa, a u normalnim uvjetima rada šalju nisku razinu. Oba senzora mogu biti istovremeno aktivna, ali zbog jednostavnosti zanemarite mogućnost da upravo tijekom prekidnog potprograma dođe novi impuls.

Glavni program cijelo vrijeme čita podatke s uvjetne jedinice VJ1. Podatke nigdje ne sprema, ali kad primi podatak s vrijednošću 12345678_{16}, onda treba zaustaviti rad programa.

U prekidnom potprogramu treba u dva brojača (brojač za senzor 1 pohranjen je na adresi 3000_{16}, a brojač za senzor 2 na adresi 3004_{16}) prebrajati koliko puta je došlo do prekoračenja tlaka na pojedinom senzoru.

```
PIOC      `EQU FFFF1000             ; adrese vanjskih jedinica
PIOD      `EQU FFFF1004
PIOIA     `EQU FFFF1008
PIOIE     `EQU FFFF100C
VJ1DATA   `EQU FFFF2000
VJ1TEST   `EQU FFFF2004
VJ1CLEAR  `EQU FFFF2004

          `ORG 0
          MOVE 10000, SP            ; inicijalizacija SP i skok u glavni
          JR GLAVNI

          `ORG 8                    ; prekidni vektor
          DW 1000

GLAVNI    MOVE %B 011111, R0        ; slanje upravljačke riječi na PIO:
          STORE R0, (PIOC)          ; OR, 1 aktiv.,mask, bit, int, ICR
          MOVE %B 11, R0            ; slanje maske
          STORE R0, (PIOC)

          MOVE %B 10010000, SR      ; dozvoli prekide

LOOP      LOAD R0, (VJ1TEST)        ; ispitaj VJ1
          OR R0, R0, R0
          JR_Z LOOP

          LOAD R0, (VJ1DATA)        ; primi podatak s VJ1
          STORE R0, (VJ1CLEAR)

          LOAD R1, (KRAJ)           ; provjeri kraj
          CMP R0, R1
          JR_NE LOOP
          HALT

KRAJ      DW 12345678               ; oznaka za kraj


          `ORG  1000                ; adresa prekidnog potprograma

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

          STORE R0, (PIOIA)         ; dojavi prihvat prekida
          LOAD R0, (PIOD)           ; pročitaj stanje senzora

          ROTR R0, 1, R0            ; ispitaj senzor 1
          JR_NC DALJE1
          LOAD R1, (3000)           ; ako je aktivan, povećaj mu brojač
          ADD R1, 1, R1
          STORE R1, (3000)

DALJE1    ROTR R0, 1, R0            ; ispitaj senzor 2
          JR_NC DALJE2
          LOAD R1, (3004)           ; ako je aktivan, povećaj mu brojač
          ADD R1, 1, R1
          STORE R1, (3004)
```

```
DALJE2    STORE R0, (PIOIE)          ; dojava kraja prekida (može i ispred RETI)

          POP R0                     ; obnova konteksta s povratkom
          MOVE R0, SR
          POP R1
          POP R0
          RETI

          `ORG 3000                  ; brojači za senzore
          DW 0
          DW 0
```

2. Međuispit iz Arhitekture računala 1 9. svibnja 2008.

Prezime i ime (štampanim slovima):_____

Matični broj: _____ **Grupa na predavanjima:** _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

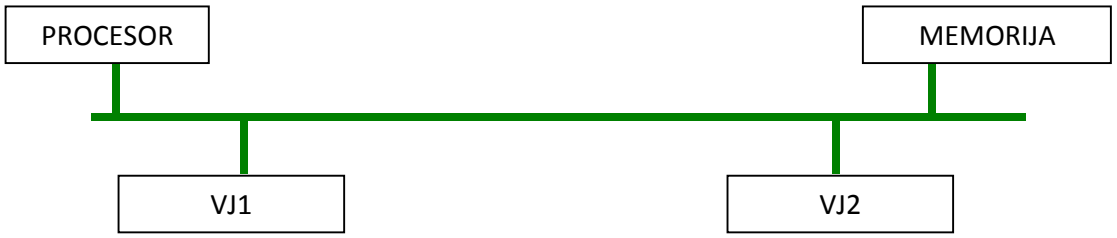
Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Zadatke 1., 2., 3., 4. i 5. rješavate na ovaj papir. Međuispit se piše 90 minuta.

1. (*0,5 boda*) Na koje priključke od FRISC-a se spajaju **vanjske jedinice** preko sabirnice tipa **spojeni-I** (*wired-and*):

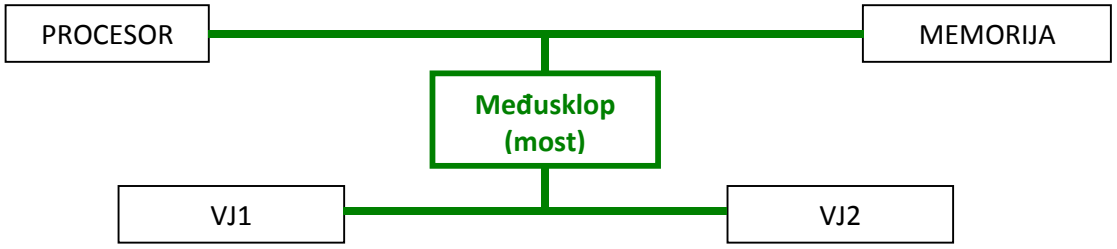
_**INT0-INT3**_____

2. (*1 bod*) U računalu treba spojiti **procesor**, **memoriju** i **dvije vanjske jedinice**. Dopunite prvu sliku tako da se spajanje izvede **zajedničkom sabirnicom (***backplane***)**, te drugu sliku tako da se spajanje izvede pomoću **međusklopa**.

a. zajednička sabirnica



b. međusklop



3. (2 *boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe CMP R1,35:

Razina dohvata:

Rastući brid CLOCK-a:

____**PC -> AR**_____

Padajući brid CLOCK-a:

____**PC +4 -> PC**_____

____**(AR) -> IR** _____

____**ext 35 i R1 -> ALU**_____

Razina izvođenja:

Rastući brid CLOCK-a:

___ **ALU izvodi oduzimanje** _____

Padajući brid CLOCK-a:

_____ **postavljanje zastavica u SR** _____

4. (*0,5 boda*) Odredite **trajanje** izvođenja sljedećeg **niza naredaba** (pretpostavite da je memorija **brza**):

ADD, STORE, JP, ADD, CMP, JR, MOVE, HALT.

1 2 2 1 1 2 1 2

Izvođenje traje _____ **12** _____ ciklusa.

5. (*1 bod*) Prednost **uvjetnog** prijenosa pred **bezuvjetnim** je:

___**nema gubitka ni uvišestručenja podataka (ili postojanje sinkronizacije)**_____.

Prednost **bezuvjetnog** prijenosa pred **uvjetnim** je:

___ **jednostavnija VJ i veća brzina (tj. nema gubitka vremena u čekanju spremnosti)** ____.

6. (*8 bodova*) Na FRISC su spojene dvije **uvjetne** jedinice VJ1 i VJ2 (na adresama FFFF 1000 i FFFF 2000), **bezuvjetna** jedinica VJ3 (adresa FFFF 3000), **CT** (adresa FFFF 4000) i **DMA** (adresa FFFF 5000).

Napišite program koji **beskonačno** prima 32-bitne podatke sa VJ1 i VJ2 (potrebno ih je **prozivati**), i **pohranjuje** ih u **blok** u memoriji. Blok počinje na lokaciji 1000_{16}. **Prilikom** svakog **pisanja** podataka u blok, na lokaciji BROJAC (900_{16}) treba **osvježiti brojač** podataka pohranjenih u blok.

CT je potrebno podesiti da **generira prekid** na INT0 svake **milisekunde** (na CNT je spojen CLOCK frekvencije 20 MHz). U prekidnom potprogramu treba **pokrenuti prijenos bloka** (popunjenog podacima s VJ1 i Vj2) na **bezuvjetnu** jedinicu VJ3 **korištenjem DMA**. DMA prijenos obavlja se **zaustavljanjem procesora** bez korištenja prekida. Nakon završetka DMA prijenosa, blok je potrebno početi **ponovno popunjavati** od početne adrese.

Prilikom pohrane podataka u blok u memoriji **ne može** doći do prepunjenja bloka. **Zanemariti** mogućnost da se prekid desi u trenutku povećavanja varijable BROJAC.

```
VJ1         `EQU  0FFFF1000
VJ1C        `EQU  0FFFF1004
VJ2         `EQU  0FFFF2000
VJ2C        `EQU  0FFFF2004
VJ3         `EQU  0FFFF3000
CTLR        `EQU  0FFFF4000
CTCR        `EQU  0FFFF4004
CTIACK      `EQU  0FFFF4008
CTIEND      `EQU  0FFFF400C
DMASRC      `EQU  0FFFF5000
DMADEST     `EQU  0FFFF5004
DMACNT      `EQU  0FFFF5008
DMACTRL     `EQU  0FFFF500C
DMASTART    `EQU  0FFFF5010
DMAIACK     `EQU  0FFFF5014

      `ORG  0
      MOVE  10000, SP   ; inicijalizacija SP-a i skok u glavni
      JP    MAIN

      `ORG  8           ; prekidni vektor
      DW    IRQ         ; ili:  DW 300

MAIN  MOVE  0,    R0
      STORE R0,   (BROJAC)    ; inicijalizacija brojača (nije nužna)

      MOVE  %D20000,R0        ; vremenska konstanta za 1 ms za CT
      STORE R0,   (CTLR)
      MOVE  %B11, R0          ; pokreni CT i neka daje prekide
      STORE R0,   (CTCR)

      MOVE  %B10010000, SR    ; omogući prekid INT0

      ; petlja prozivanja
PETLJA      LOAD  R0,   (VJ1C)      ; provjera spremnosti VJ1
      OR    R0,   R0,   R0
      JP_Z  DALJE

      ; posluživanje VJ1
      LOAD  R0,   (VJ1)       ; učitaj podatak sa VJ1
      STORE R0,   (VJ1C)      ; briši spremnost od VJ1

      LOAD  R1,   (BROJAC)    ; dohvati brojač
      SHL   R1,   2,    R2    ; izračunaj adresu (1 podatak = 4 bajta)

      STORE R0,   (R2 + BLOK) ; i spremi na nju dohvaćeni podatak

      ADD   R1,   1,    R1    ; povećaj brojač
      STORE R1,   (BROJAC)

DALJE LOAD  R0,   (VJ2C)      ; provjera spremnosti VJ2
```

```
      OR    R0,   R0,   R0
      JP_Z  PETLJA

      ; posluživanje VJ2 - analogno gornjem posluživanju za VJ1
      LOAD  R0,   (VJ2)
      STORE R0,   (VJ2C)
      LOAD  R1,   (BROJAC)
      SHL   R1,   2,    R2
      STORE R0,   (R2 + BLOK)
      ADD   R1,   1,    R1
      STORE R1,   (BROJAC)

      JP    PETLJA


; prekidni potprogram
      `ORG  300
IRQ   PUSH  R0    ; premanje konteksta (ovaj potprogram ne mijenja SR)

      STORE R0,   (CTIACK)    ; potvrda prihvata prekida CT-u

      MOVE  BLOK, R0          ; inicijaliziraj početnu adresu za DMA
      STORE R0,   (DMASRC)

      MOVE  VJ3,  R0          ; inicijaliziraj odredišnu adresu za DMA
      STORE R0,   (DMADEST)

      LOAD  R0,   (BROJAC)    ; zadaj broj podataka za DMA
      STORE R0,   (DMACNT)

      MOVE  %B1000,     R0    ; način rada DMA: iz MEM u VJ, bez prekida, halting
      STORE R0,   (DMACTRL)
      STORE R0,   (DMASTART)  ; pokreni DMA


      MOVE  0,    R0          ; vrati BROJAC na 0
      STORE R0,   (BROJAC)

      STORE R0,   (CTIEND)    ; dojavi kraj obrade prekida CT-u

      POP   R0          ; obnova konteksta

      RETI        ; povratak iz prek.potp.

; podaci
BROJAC      DW 1

      `ORG  1000  ; mjesto za blok
BLOK  `DS ...
```

7. (*7 bodova*) Na FRISC su spojena **2 CT sklopa** CT1 i CT2 , **PIO** i **bezuvjetna** jedinica VJ. Adrese vanjskih jedinica odaberite sami.

PIO sklop spojen je na prekidni priključak **INT3 (Pozor: PIO nema priključak IACK)**, a **CT2** na priključak **INT2**.

Na **donjih 5** linija PIO sklopa spojeni su senzori aktivni u **stanju 1**. Potrebno je podesiti PIO da generira prekid INT3 kad je **bilo koji** senzor aktivan.

U **prekidnom potprogramu** za obradu **prekida** generiranog PIO-m (**INT3**) treba **podesiti CT** sklopove tako da generiraju prekid na INT2 **nakon** isteka **5 sekundi od trenutka aktiviranja senzora**. Na CNT od CT1 je spojen CLOCK frekvencije 20 MHz.

U **prekidnom potprogramu** za obradu prekida nakon 5 sekundnog kašnjenja (**INT2)** treba na **bezuvjetnu** jedinicu VJ poslati broj 33FF. To treba napraviti samo jednom nakon isteka 5 sekundi od aktiviranja senzora, a ne svakih 5 sekundi.

Glavni program nakon potrebnih inicijalizacija treba izvoditi **petlju**.

Pretpostavite da senzori izazivaju prekide relativno rijetko, tj. sigurno ih neće izazvati prije isteka razdoblja od 5 sekundi. **Skicirajte ili opišite** način međusobnog spajanja CT sklopova i spajanja na INT2 od FRISC-a.

```
CT1LR       `EQU  0FFFF1000
CT1CR       `EQU  0FFFF1004
CT2LR       `EQU  0FFFF2000
CT2CR       `EQU  0FFFF2004
CT2IACK     `EQU  0FFFF2008
CT2IEND     `EQU  0FFFF200C
PIO         `EQU  0FFFF3000
PIOD        `EQU  0FFFF3004
PIOIACK     `EQU  0FFFF3008
PIOIEND     `EQU  0FFFF300C
VJ          `EQU  0FFFF4000

      `ORG  0
      MOVE  10000,      SP
      JP    MAIN

      `ORG  8           ; prekidni vektor za INT2
      `DW   500

      ; prekidni potprogram za nemaskirajući INT3
      `ORG  %D12

NMI   PUSH  R0          ; spremanje konteksta
      MOVE  SR, R0
      PUSH  R0

      STORE R0, (PIOIACK) ; treba programski iack jer PIO nema IACK liniju

      MOVE  %D10000,    R0    ; vremnske konstante (dva puta po 10000)
      STORE R0,   (CT1LR)
      STORE R0,   (CT2LR)

      MOVE  %B10, R0          ; CT1 broji, ali bez prekida
      STORE R0,   (CT1CR)

      MOVE  %B11, R0          ; CT2 broji i generira prekid
      STORE R0,   (CT2CR)

      POP   R0          ; obnovi kontekst
      MOVE  R0,SR
      POP   R0
      STORE R0, (PIOIEND)     ; dojava kraja obrade prekida
      RETN        ; povratak iz NMI
```

```
; GLAVNI program

MAIN  MOVE  %B011111,   R0    ; upravljačka riječ: OR, aktiv=1, maska slijedi; INT
      STORE R0,   (PIO)

      STORE R0,   (PIO)       ; slanje maske, slučajno je ista kao upravljačka riječ

      MOVE  %B11000000, SR    ; onogući INT2

PETLJA      JP    PETLJA      ; "glavni program"



; prekidni program za CT2

      `ORG 500
IRQ   PUSH  R0                ; spremi kontekst (SR se ne mijenja)

      STORE R0,   (CT2IACK)   ; dojavi CT2-u da je prekid prihvaćen

      MOVE  0, R0             ; spriječi nove prekide sa CT2
      STORE R0,(CT2CR)        ; (nije nužno zaustavljati CT1)

      MOVE  33FF, R0          ; pošalji 33FF na VJ
      STORE R0, (VJ)

      STORE R0,   (CT2IEND)   ; dojavi CT2-u da je prekid obrađen

      POP   R0                ; obnovi kontekst

      RETI        ; povratak iz maskirajućeg prekida
```

IZLAZ (priključak ZC) od CT1 treba spojiti na ULAZ (priključak CNT) od CT2, a priključak INT od CT2 treba spojiti na priključak INT2 od FRISC-a

2. Međuispit iz Arhitekture računala 1 13. svibnja 2009.

Prezime i ime (tiskanim slovima): _____

Matični broj: _____ Grupa na predavanjima: _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

Dozvoljeno je koristiti isključivo službeni šalabahter (popis naredaba FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Zadatke 1., 2., 3., 4. i 5. rješavate na ovaj papir. Međuispit se piše 90 minuta.

1. (*1 bod*) Procesor FRISC se sinkronizira s DMA-jedinicom pomoću:

_____ (navedite ime FRISC-ovog priključka) čiji smjer je _____,

_____ (navedite ime FRISC-ovog priključka) čiji smjer je _____.

2. (*2 boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **STORE R1, (3000)**:

Razina dohvata:

Rastući brid CLOCK-a:

_____

Padajući brid CLOCK-a:

_____

_____

_____

_____

Razina izvođenja:

Rastući brid CLOCK-a:

_____

_____

Padajući brid CLOCK-a:

_____

_____

3. (*1,5 bod*) Odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**):

```
     `ORG 0
      MOVE   4,  R0
POC   SUB    R0, 2, R0
      JR_NZ  POC
      STORE  R0, (1000)
      HALT
```

Izvođenje traje _____ ciklusa. Kraj **svake** naredbe napišite **koliko** puta se izvodi i koliko je njeno **trajanje** u ciklusima.

4. (*0,5 boda*) **Uvjetna** vanjska jedinica u odnosu na **bezuvjetnu** vanjsku jedinicu u svojoj unutrašnjosti mora imati _____ (koji dio). **Uvjetna** vanjska jedinica u odnosu na **bezuvjetnu** vanjsku jedinicu mora imati dodatne priključke za spajanje prema vanjskom svijetu. Ovi priključci nazivaju se priključcima za _____.

OKRENI!

5. (*7 bodova*) Na FRISC su spojeni **DMA** (adresa FFFF 1000_{16}), **disk** (adresa FFFF 2000_{16}) i **CT** (spojen na INT3, CT nema priključak IACK, adresa FFFF 3000_{16}).

Program treba slijedno pokretati **DMA** (zaustavljanje procesora, ne postavlja prekid) **svakih 1 sekundu** pomoću **CT** sklopa (na CT-ov ulaz CNT spojen je signal frekvencije 25 kHz). U svakom DMA prijenosu se prenosi **po jedan blok** od 14_{16} (20_{10}) 32-bitnih podataka **s diska u memoriju**, počevši slijedno od memorijske lokacije 1000_{16} na dalje (1. blok na 1000_{16}, 2. blok na 1050_{16}, itd.). Nakon prijenosa 100_{10} blokova treba zaustaviti procesor. Nakon inicijalizacije vanjskih jedinica, glavni program treba cijelo vrijeme izvoditi praznu petlju.
Sa diska se podaci čitaju bezuvjetno, a disk u sebi ima međuspremnik za podatke kapaciteta 20_{10} riječi te zauzima 20_{10} 32-bitnih lokacija počevši od adrese FFFF 2000_{16}.

Potprogrami trebaju čuvati stanje registara osim onih koji se koriste za prijenos podataka.

6. (*8 bodova*) Na FRISC su spojene uvjetne ulazne vanjske jedinice VJ1 (adresa FFFF 1000_{16}) i VJ2 (adresa FFFF 2000_{16}), uvjetna izlazna vanjska jedinica VJ3 (adresa FFFF 3000_{16}) i PIO u prekidnom načinu rada (spojen na INT1, adresa FFFF 4000_{16}).

Napisati glavni program koji **beskonačno** prima podatke s VJ1 i VJ2 (jedinice su međusobno **nezavisne**). Svaki primljeni podatak treba **poslati** na VJ3. Ako je podatak **djeljiv s 4**, treba povećati brojač prenesenih podataka BROJAC4 pohranjen na memorijskoj lokaciji 1000_{16}. *Na primjer, ako su preneseni podaci 1_{10}, 24_{10}, 5_{10}, 6_{10}, 8_{10} brojač sadrži vrijednost 2 jer su samo 24_{10} i 8_{10} djeljivi s 4.*

Napisati **prekidni potprogram** za obradu **prekida** PIO, koji na PIO šalje sadržaj brojača BROJAC4 i ponovno postavlja brojač na 0. Pretpostavlja se da sadržaj brojača neće prijeći opseg od 8 bita.

Potprogrami trebaju čuvati stanje registara osim onih koji se koriste za prijenos podataka.

RJEŠENJA:

1. (*1 bod*) Procesor FRISC se sinkronizira s DMA-jedinicom pomoću:
__**BREQ**_____ (navedite ime FRISC-ovog priključka) čiji smjer je ___**ulazni**_____,
__**BACK**_____ (navedite ime FRISC-ovog priključka) čiji smjer je ___**izlazni**_____.

2. (2 *boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe STORE R1,(3000):

Razina dohvata:

Rastući brid CLOCK-a:

____**PC -> AR**_____

Padajući brid CLOCK-a:

____**PC +4 -> PC**_____

____**(AR) -> IR, dekodiranje** _____

____**ext 3000** _____

_ **onemogući dohvat u sljedećem ciklusu** _

Razina izvođenja:

Rastući brid CLOCK-a:

___ **ext 3000 -> AR** _____

___**R1 -> DR**_____

Padajući brid CLOCK-a:

__**DR -> (AR)**_____

__ **omogući dohvat u sljedećem ciklusu** ___

3. (1,5 *bod*) Odredite **trajanje** izvođenja sljedećeg **programskog odsječka** (pretpostavite da je memorija **brza**):

```
      `ORG 0                          0 ☺
       MOVE   4, R0                   1 x 1 = 1
POC    SUB    R0, 2, R0               2 x 1 = 2
       JR_NZ  POC                     2 x 2 = 4
       STORE  R0, (1000)              1 x 2 = 2
       HALT                                  1 x 2 = 2
```

Izvođenje traje _____ **11** _____ ciklusa . Mora se vidjeti postupak računanja, tj. kraj svake naredbe mora se vidjeti koliko puta se izvodi i koliko je njeno trajanje

4. (0,5 *boda*) **Uvjetna** vanjska jedinica o odnosu na **bezuvjetnu** vanjsku jedinicu u svojoj unutrašnjosti mora imati _____**bistabil stanja (ili status bistabil)**_____ (koji dio). **Uvjetna** vanjska jedinica o odnosu na **bezuvjetnu** vanjsku jedinicu mora imati dodatne priključke za spajanje prema vanjskom svijetu. Ovi priključci nazivaju se priključcima za _____**sinkronizaciju (ili rukovanje)**_____.

5. (*7 bodova*) Na FRISC su spojeni **DMA** (adresa FFFF 1000_{16}), **disk** (adresa FFFF 2000_{16}) i **CT** (spojen na INT3, CT nema priključak IACK, adresa FFFF 3000_{16}).

Program treba slijedno pokretati **DMA** (zaustavljanje procesora, ne postavlja prekid) **svakih 1 sekundu** pomoću **CT** sklopa (na CT-ov ulaz CNT spojen je signal frekvencije 25 kHz). U svakom DMA prijenosu se prenosi **po jedan blok** od 14_{16} (20_{10}) 32-bitnih podataka **s diska u memoriju**, počevši slijedno od memorijske lokacije 1000_{16} na dalje (1. blok na 1000_{16}, 2. blok na 1050_{16}, itd.). Nakon prijenosa 100_{10} blokova treba zaustaviti procesor. Nakon inicijalizacije vanjskih jedinica, glavni program treba cijelo vrijeme izvoditi praznu petlju.
Sa diska se podaci čitaju bezuvjetno, a disk u sebi ima međuspremnik za podatke kapaciteta 20_{10} riječi te zauzima 20_{10} 32-bitnih lokacija počevši od adrese FFFF 2000_{16}.

Potprogrami trebaju čuvati stanje registara osim onih koji se koriste za prijenos podataka.

```
DMASRC          `EQU  0FFFF1000
DMADEST     `EQU  0FFFF1004
DMACNT          `EQU  0FFFF1008
DMACTRL     `EQU  0FFFF100C
DMASTART    `EQU  0FFFF1010
DMAIACK     `EQU  0FFFF1014             ; konstante DMA

DISK        `EQU  0FFFF2000

CTLR        `EQU  0FFFF3000
CTCR        `EQU  0FFFF3004
CTIACK          `EQU  0FFFF3008
CTIEND          `EQU  0FFFF300C             ; konstante DISK i CT

                `ORG 0
                JP GLAVNI                   ; skok na glavni

                `ORG 0C                     ; NMI

                JP NMI                          ; nije  vektor nego skok ili call


GLAVNI          MOVE  10000, SP             ; stog

                LOAD  R0, (KONST)       ; vremenska konstanta (ok i MOVE)
                STORE R0, (CTLR)        ; postavi CTLR

                MOVE  %B11, R0              ; broji, prekid
                STORE R0, (CTCR)        ; pokreni CT

                ; postavljanje SR nije potrebno - INT3

PETLJA          JP PETLJA                   ; prazna petlja



                ; prekidni potprogram
NMI             PUSH  R0                    ; moraju cuvati kontekst..
                MOVE  SR, R0                ; ..bez obzira sto se ne koristi..
                PUSH  R0                    ; ..niti jedan registar u glavnom.

                STORE R0, (CTIACK)          ; rucni IACK

                MOVE  DISK, R0              ; pocetna adresa - disk
                STORE R0, (DMASRC)          ; postavi pocetnu adresu

                LOAD  R0, (BLOK)        ; ucitaj sljedecu adresu bloka
                STORE R0, (DMADEST)             ; odredisna adresa

                ADD         R0, %D80, R0            ; pomakni blok

                STORE R0, (BLOK)        ; pohrani sljedecu adresu
```

```
                MOVE  R0, %D20                  ; velicina bloka
                STORE R0, (DMACNT)              ; postavi brojac DMA

                MOVE  %B0000, R0        ; mem->mem, stop, noint
                STORE R0, (DMACTRL)             ; kontrolna rijec

                STORE R0, (DMASTART)    ; pokreni DMA

                ; NOP                                   ; prijenos

                LOAD  R0, (BROJAC)
                SUB     R0, 1, R0               ; smanji brojac

                JP_Z  KRAJ

                STORE R0, (BROJAC)              ; spremi brojac

                STORE R0, (CTIEND)              ; zavrsi prekid

                POP   R0                                ; vracanje konteksta

                MOVE  R0, SR
                POP     R0

                RETN                                    ; vracanje iz potprograma

KRAJ        HALT                                ; zaustavi procesor


                ; varijable
BLOK        DW    1000                          ; poc. adr. bloka
BROJAC          DW %D 100                       ; brojac blokova

KONST       DW %D 25000                 ; vremenska konstanta za CT
```

6. (*8 bodova*) Na FRISC su spojene uvjetne ulazne vanjske jedinice VJ1 (adresa FFFF 1000_{16}) i VJ2 (adresa FFFF 2000_{16}), uvjetna izlazna vanjska jedinica VJ3 (adresa FFFF 3000_{16}) i PIO u prekidnom načinu rada (spojen na INT1, adresa FFFF 4000_{16}).

Napisati glavni program koji **beskonačno** prima podatke s VJ1 i VJ2 (jedinice su međusobno **nezavisne**). Svaki primljeni podatak treba **poslati** na VJ3. Ako je podatak **djeljiv s 4**, treba povećati brojač prenesenih podataka BROJAC4 pohranjen na memorijskoj lokaciji 1000_{16}. *Na primjer, ako su preneseni podaci 1_{10}, 24_{10}, 5_{10}, 6_{10}, 8_{10} brojač sadrži vrijednost 2 jer su samo 24_{10} i 8_{10} djeljivi s 4.*

Napisati **prekidni potprogram** za obradu **prekida** PIO, koji na PIO šalje sadržaj brojača BROJAC4 i ponovno postavlja brojač na 0. Pretpostavlja se da sadržaj brojača neće prijeći opseg od 8 bita.

Potprogrami trebaju čuvati stanje registara osim onih koji se koriste za prijenos podataka.

```
VJ1             `EQU  0FFFF1000
VJ1CL       `EQU  0FFFF1004

VJ2             `EQU  0FFFF2000
VJ2CL       `EQU  0FFFF2004

VJ3             `EQU  0FFFF3000
VJ3CL       `EQU  0FFFF3004             ; sve tri VJ1, VJ2, VJ3

PIOC        `EQU  0FFFF4000
PIOD        `EQU  0FFFF4004
PIOIACK     `EQU  0FFFF4008
PIOIEND     `EQU  0FFFF400C             ; konstante za PIO

            `ORG 0
            JP GLAVNI                           ; skok na glavni

            `ORG 8                                      ; prekidni vektor
            DW    IRQ                           ; adresa


GLAVNI      MOVE  10000, SP             ; stog

            MOVE  %B010, R0             ; izlazni, prekidi, bajt
            STORE R0, (PIOC)        ; podesi PIO

            MOVE  %B10100000, SR    ; GIE, EINT1

            ; ispravno prozivanje (slucaj da je VJ1 puno brza od VJ2)

PET1        LOAD  R0, (VJ1CL)       ; spremna prva?
            CMP         R0, 0                   ; usporedba i JP
            JP_EQ PET2                  ; JP_EQ ili JP_Z

            LOAD  R1, (VJ1)             ; ucitaj podatak u R1
            STORE R1, (VJ1CL)       ; obrisi spremnost

            CALL  OBRADI                        ; posalji R1 na VJ3

PET2        LOAD  R0, (VJ2CL)       ; spremna druga?
            CMP         R0, 0                   ; usporedba i JP
            JP_EQ PET1                  ; JP_EQ ili JP_Z

            LOAD  R1, (VJ2)             ; ucitaj podatak u R1
            STORE R1, (VJ2CL)       ; obrisi spremnost

            CALL  OBRADI                        ; posalji R1 na VJ3

            JP          PET1                    ; petlja
```

```
                    ; potprogram za prijenos na VJ3
                    ; R1 = podatak
OBRADI              PUSH   R0                       ; kontekst

PET3        LOAD    R0, (VJ3CL) ; spremna treca?
                    CMP          R0, 0              ; usporedba i JP
                    JP_EQ PET3                      ; JP_EQ ili JP_Z

                    STORE  R1, (VJ3)                ; posalji podatak
                    STORE  R1, (VJ3CL) ; obrisi spremnost

                    AND          R1, %B11, R1       ; ispitaj djeljivost
                    JP_NZ NAZAD                     ; nije djeljiv s 4

                    LOAD   R1, (BROJAC4)
                    ADD            R1, 1, R1
                    STORE  R1, (BROJAC4)            ; povecaj brojac

NAZAD       POP            R0                       ; kontekst
                    RET                                     ; povratak


                    ; kraj


                    ; prekidni potprogram
IRQ                 PUSH   R0                       ; pohrani kontekst
                    MOVE   SR, R0                   ; nije potrebno, LOAD, STORE i MOVE...
                    PUSH   R0                       ; ...ne mijenjaju zastavice

                    STORE  R0, (PIOIACK)            ; potvrdi prekid

                    LOAD   R0, (BROJAC4)            ; ucitaj brojac
                    STORE  R0, (PIOD)   ; posalji na PIO

                    MOVE   0, R0                    ; postavi na 0
                    STORE  R0, (BROJAC4)            ; pohrani

                    STORE  R0, (PIOIEND)            ; zavrsi prekid

                    POP            R0                       ; vrati kontekst
                    MOVE   R0, SR                   ; nije potrebno, LOAD, STORE i MOVE...
                    POP            R0                       ; ...ne mijenjaju zastavice

                    RETI                            ; povratak iz prekida

                    `ORG   1000
BROJAC4     DW             0                        ; varijabla (a ne konstanta)
```

2. međuispit iz ARH 1 18. svibnja 2010.

Prezime i ime (velikim 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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Prvi zadatak rješavati na ovaj papir. Međuispit traje 90 minuta.

1a. *(2 boda)* Kad FRISC započinje sa čitanjem iz memorije, na priključku READ postavit će stanje ____, a na priključak WRITE stanje_____. Na sabirnicu adresa, FRISC postavlja stanje iz registra _____. Ako je memorija spora, mora postaviti stanje _____ na priključak _____. Memorija to mora napraviti prije (kojeg trenutka) _____. U ciklusu čitanja, sabirnicom adresa upravlja _____, a sabirnicom podataka upravlja _____. U ciklusu pisanja, sabirnicom adresa upravlja _____, a sabirnicom podataka upravlja _____.

1b. (*2 boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe JP_NV 500:

Razina dohvata:
Rastući brid CLOCK-a:
_____
Padajući brid CLOCK-a:
_____
_____
_____
_____
_____

Razina izvođenja:
Rastući brid CLOCK-a:
Padajući brid CLOCK-a:
_____
_____

1c. (*2 boda*) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-PIO zovu se. _____. Ovi priključci koriste se u načinima rada (nabrojite kojim): _____. FRISC-PIO **ne može** postati spreman u načinu (načinima) rada: _____.
Osim registra maske, na prvoj adresi sklopa PIO nalaze se upravljački registri: _____. Kad šaljemo upravljačku riječ na tu adresu, PIO zna u koji registar je želimo upisati na temelju (čega?): _____ _____. Ako želimo da PIO (spojen na adresi FFFF4400) generira prekid kad se na svih 5 najnižih bitova postave jedinice, onda ga inicijaliziramo tako da pošaljemo podatak _____ na adresu _____ i zatim podatak _____ na adresu _____.

2. *(6 bodova)* Na FRISC su spojene tri vanjske jedinice: PIO, uvjetna jedinica ZASLON i bezuvjetna jedinica ZVUCNIK. Na PIO je spojena tipkovnica, a PIO je spojen na INT0. Svaki puta kada je na tipkovnici pritisnuta tipka ona šalje PIO-sklopu 8-bitni ASCII kôd pritisnute tipke. Svi potprogrami trebaju čuvati kontekst.

Napišite program koji pomoću prekida prima znakove s PIO-sklopa i šalje ih na zaslon. ZASLON prima znakove u ASCII-formatu i prikazuje ih. Pretpostavite da je jedinica ZASLON vrlo brza u odnosu na tipkovnicu.

Samo ako je sa PIO primljen ASCII-kod znaka BEL (zvonce), čija vrijednost je 7, onda ne treba slati znak na ZASLON, nego treba aktivirati ZVUCNIK slanjem bilo kojeg podatka. Glavni program izvodi praznu petlju.

3. (*8 bodova*) Na FRISC su spojeni DMA i dva sklopa CT: CT1 i CT2. Na priključak CNT sklopa CT1 spojen je signal CLOCK frekvencije 1 GHz. CT1 spojen je na INT1, a CT2 na INT2.

U memoriji postoji izvorišni blok 32-bitnih podataka koji počinje od adrese 1000_{16} i sadrži 100_{10} podataka. Od adrese 2000_{16} se nalazi 1000_{10} 32-bitnih lokacija odredišnog bloka podataka. Svaki podatak izvorišnog bloka treba kopirati na 10_{10} slijednih 32-bitnih lokacija odredišnog bloka.

Svakih 2,5 sekunde (kašnjenje ostvariti raspoloživim CT-ovima pomoću prekida) treba pomoću DMA kopirati sljedeći podatak izvorišnog bloka u 10 podataka odredišnog bloka. DMA treba raditi krađom ciklusa i pomoću INT3 treba dojaviti da je blok prenesen (pretpostavite da DMA-sklop ima ulaz IACK). U prekidnom potprogramu za DMA treba prebrajati koliko izvorišnih podataka je preneseno. Pretpostavka je da je DMA prijenos 10 podataka puno brži od 2,5 sekunde. Svi potprogrami trebaju čuvati kontekst.

Glavni program treba ispitivati koliko je izvorišnih podataka preneseno i treba zaustaviti procesor nakon što su kopirani svi podaci izvorišnog bloka. Ako CT-ovi trebaju biti međusobno spojeni, napišite ili skicirajte način spajanja.

RJEŠENJA

1a. *(2 boda)* Kad FRISC započinje sa čitanjem iz memorije, na priključku READ postavit će stanje _**0**_, a na priključak WRITE stanje___**1**___. Na sabirnicu adresa, FRISC postavlja stanje iz registra __**AR**___. Ako je memorija spora, mora postaviti stanje ____**0**____ na priključak ___**WAIT**___. Memorija to mora napraviti prije (kojeg trenutka) ___**padajućeg brida CLOCK-a**_____. U ciklusu čitanja, sabirnicom adresa upravlja ___**FRISC (ili procesor)**___, a sabirnicom podataka upravlja __**memorija**___. U ciklusu pisanja, sabirnicom adresa upravlja __ **FRISC** ___, a sabirnicom podataka upravlja ___ **FRISC** _____.

1b. (*2 boda*) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe JP_NV 500:

Razina dohvata:
- Rastući brid CLOCK-a:
  - _____ **PC -> AR** _____
- Padajući brid CLOCK-a:
  - _____ **PC+4 -> PC** _____
  - _____ **(AR) -> IR, dekodiranje** _____
  - _____ **ispitivanje uvjeta V=0** _____
  - _____**ako je V=0: ext 500** _____
  - _**onemogući dohvat u sljedećem ciklusu**__

Razina izvođenja:
- Rastući brid CLOCK-a:
- Padajući brid CLOCK-a:
  - __ **ako je V=0: ext 500 -> PC**_____
  - _ **omogući dohvat u sljedećem ciklusu** _

1c. (*2 boda*) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-PIO zovu se. ____**READY i STROBE**_____. Ovi priključci koriste se u načinima rada (nabrojite kojim): ___**ulazni i izlazni**_____. FRISC-PIO **ne može** postati spreman u načinu (načinima) rada: _____**postavljanje bitova**_____.
Osim registra maske, na prvoj adresi sklopa PIO nalaze se upravljački registri: __**ICR i OCR**__. Kad šaljemo upravljačku riječ na tu adresu, PIO zna u koji registar je želimo upisati na temelju (čega?): __**najnižeg bita poslane upravljačke riječi (ili poslanog podatka)** _____. Ako želimo da PIO (spojen na adresi FFFF4400) generira prekid kad se na svih 5 najnižih bitova postave jedinice, onda ga inicijaliziramo tako da pošaljemo podatak __**00111111_2**___ na adresu _ **FFFF4400**__ i zatim podatak _____**00011111_2**_____ na adresu __ **FFFF4400**___.

RJEŠENJE 2. ZADATAK

```
        `ORG   0
PIOC            `EQU   0FFFF1000
PIOD            `EQU   0FFFF1004
PIOIACK         `EQU   0FFFF1008
PIOIEND         `EQU   0FFFF100C
ZASLON          `EQU   0FFFF2000
ZASLONI         `EQU   0FFFF2004
ZASLONB         `EQU   0FFFF2004
ZVUCNIK         `EQU   0FFFF3000

        MOVE   10000, SP          ; stog
        JP     GLAVNI             ; skok na glavni
        `ORG   8                  ; adresa prek. vektora
        DW     1000               ; prekidni vektor

GLAVNI  MOVE   %B011, R0          ; ulazni, prekid
        STORE  R0, (PIOC)         ; upis u PIO ICR
        MOVE   %B10010000, SR     ; omogući prekid INT0
PET     JR     PET                ; prazna petlja
        HALT                      ; zaustavi proc.

        `ORG   1000               ; prekidni potprogram PIO
        PUSH   R0                 ; kontekst
        MOVE   SR, R0             ; statusni reg
        PUSH   R0
        PUSH   R1
        STORE  R0, (PIOIACK)      ; obriši spremnost
        LOAD   R1, (PIOD)         ; učitaj kôd tipke
        CMP    R1, 7              ; bell?
        JP_NE  PETLJA             ; ako nije bell dalje
        STORE R1, (ZVUCNIK)       ; zazvoni
        JR     VAN                ; skok van
PETLJA  LOAD   R0, (ZASLONI)      ; spreman zaslon
        OR     R0, R0, R0         ; ispitaj
        JR_Z   PETLJA             ; vrti petlju
        STORE  R1, (ZASLON)       ; pošalji na zaslon
        STORE  R0, (ZASLONB)      ; briši
VAN     STORE  R0, (PIOIEND)      ; dojava kraja
        POP    R1
        POP    R0
        MOVE   R0, SR             ; statusni reg
        POP    R0                 ; vrati kontekst
        RETI                      ; povratak
```

RJEŠENJE 3. ZADATAK

(CT1LR) 50.000 * (CT2LR) 50.000 / (freq) 1.000.000.000 = 2.500.000.000 / 1.000.000.000 = 2,5

IZLAZ (priključak ZC) CT1 spojiti na ULAZ (priključak CNT) CT2, a priključak INT CT2 spojiti na priključak INT2 FRISC-a

```
        `ORG   0
CT1LR           `EQU   0FFFF1000
CT1CR           `EQU   0FFFF1004
CT2LR           `EQU   0FFFF2000
CT2CR           `EQU   0FFFF2004
CT2IACK         `EQU   0FFFF2008
CT2IEND         `EQU   0FFFF200C
DMASRC          `EQU   0FFFF5000
DMADEST         `EQU   0FFFF5004
DMACNT          `EQU   0FFFF5008
DMACTRL         `EQU   0FFFF500C
DMASTART        `EQU   0FFFF5010
DMAIACK         `EQU   0FFFF5014

        MOVE   10000, SP          ; stog
        JP     GLAVNI             ; skok na glavni
        `ORG   8                  ; adresa prek. vektora
        DW     500                ; prekidni vektor

        `ORG   %D12               ; prekidni potprogram DMA
        PUSH   R0                 ; kontekst
        MOVE   SR, R0
        PUSH   R0
        LOAD   R0,    (BROJAC)    ; učitaj BROJAC
        ADD    R0, 1, R0          ; povećaj za 1
        STORE  R0,    (BROJAC)    ; učitaj BROJAC
        POP    R0
        MOVE   R0, SR
        POP    R0                 ; kontekst
        RETN                      ; povratak

GLAVNI  MOVE   %D50000, R0        ; 50000
        STORE  R0,    (CT1LR)     ; upis u LR CT1
        STORE  R0,    (CT2LR)     ; upis u LR CT2
        MOVE   %B10,  R0          ; CT1 broji
        STORE  R0,    (CT1CR)     ; upis u CR CT1
        MOVE   %B11,  R0          ; CT2 broji i prekid
        STORE  R0,    (CT2CR)     ; upis u CR CT2
        MOVE   %B11000000, SR     ; omogući prekid INT2
PETLJA  LOAD   R2,    (BROJAC)    ; učitaj BROJAC
        CMP    R2, %D100          ; izbrojeno 100
        JP_NZ  PETLJA             ; vrti petlju
        HALT                      ; zaustavi proc.
BROJAC  DW     0                  ; brojač

        `ORG   500                ; prekidni potprogram CT
        PUSH   R0                 ; kontekst
        PUSH   R1                 ;
        PUSH   R2                 ;
        MOVE   SR, R2             ; zastavice
        PUSH   R2                 ;
        LOAD   R0,    (IZVOR)     ; učitaj tren. varijable
        LOAD   R1,    (ODRED)     ;
        STORE  R0,    (CT2IACK)   ; obriši spremnost
        STORE  R0,    (DMASRC)    ; adresa podatka
        ADD    R0, 4, R0          ; pomak adrese podatka
        STORE  R1,    (DMADEST)   ; adresa odr. bloka
        ADD    R1, %D40, R1       ; pomak adrese odr. bloka
        STORE  R0,    (IZVOR)     ; spremi varijable
        STORE  R1,    (ODRED)     ;
        MOVE   %D10, R2           ; brojač podataka
```

```
        STORE  R2,    (DMACNT)     ; upis brojača
        MOVE   %B0111, R2          ; riječ za DMA
        STORE  R2,    (DMACTRL)    ; pohrani upr. riječ
        STORE  R0,    (DMASTART)   ; pokreni DMA
        STORE  R0,    (CT2IEND)    ; dojava kraja
        POP    R2
        MOVE   R2,SR
        POP    R2
        POP    R0                  ; kontekst
        RETI                       ; povratak

IZVOR   DW     1000                ; početne vrijednosti var
ODRED   DW     2000
```

2. međuispit iz Arhitekture računala 1 — 9. svibnja 2011.

Grupa na predavanjima: _____

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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Međuispit traje 95 minuta.

1. (1,5 bodova) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **MOVE 50, R1**:

Razina dohvata:

Rastući brid CLOCK-a:

_____

Padajući brid CLOCK-a:

_____

_____

_____

Razina izvođenja:

Rastući brid CLOCK-a:

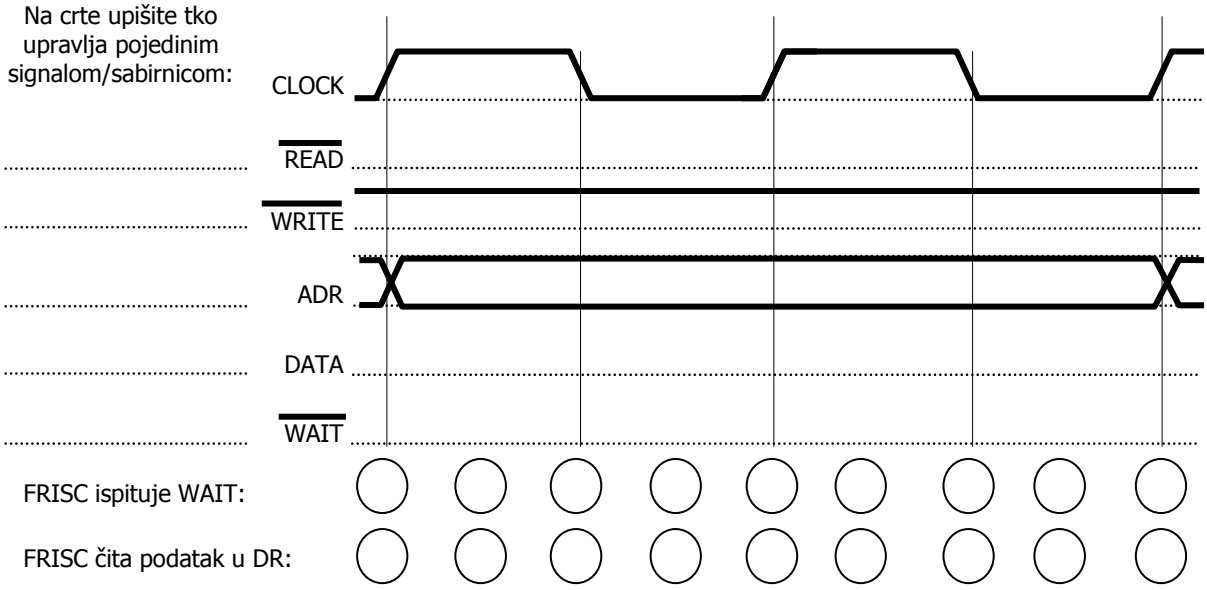
_____

Padajući brid CLOCK-a:

_____

2. (3,5 bodova) **Nacrtajte stanja na sabirnicama** FRISC-a za čitanje podatka iz memorije s jednim stanjem čekanja. Na crte s lijeve strane **upišite tko upravlja** pojedinom sabirnicom/signalom (FRISC ili memorija). **Stavite oznake** "X" u one kružiće u kojima:

- FRISC ispituje signal WAIT da utvrdi je li memorija obavila traženu operaciju (gornji red)
- FRISC čita podatak sa sabirnice podataka u interni registar DR (donji red)



3. (1,5 bodova) Odredite trajanje izvođenja sljedećeg programskog odsječka (pretpostavite da je memorija brza). Mora se vidjeti postupak računanja, tj. kraj svake naredbe napisati koliko puta se izvodi i koliko je njeno trajanje.

```
            `ORG 0
            JP GLAVNI
OSAM        `EQU 8
GLAVNI      MOVE OSAM, R0
PETLJA      ROTR R0, 2, R0
            JR_NC PETLJA
            STORE R0, (100)
            HALT
```

Izvođenje traje _____ ciklusa.

OKRENI PAPIR!

4. (*6 bodova*) Na FRISC su spojeni **DMA** (adresa FFFF 1000), **bezuvjetna** vanjska jedinica BVJ (adresa FFFF 2000) i **uvjetne** vanjske jedinice UVJ1 i UVJ2 (adrese FFFF 3000 i FFFF 4000).

Napišite program koji treba prenijeti 100_{16} 32-bitnih podataka s BVJ u blok memorije od adrese 1000_{16}, pomoću **DMA prijenosa**. DMA treba raditi krađom ciklusa i pomoću prekida na INT0 dojaviti da je prijenos završen. Tijekom DMA prijenosa treba **prenositi podatke** iz UVJ1 na UVJ2 te u memorijskoj lokaciji BROJAC **prebrajati** koliko je podataka preneseno.

Nakon dovršenog DMA prijenosa, treba **prestati prenositi podatke** između uvjetnih vanjskih jedinica te početi izvoditi praznu petlju u glavnom programu. Potprogram treba čuvati kontekst.

5. (7,5 *bodova*) Na FRISC je spojen alarm za bicikl koji se sastoji od **uvjetne vanjske jedinice** spojene na senzor vibracija – detektira pokušaj krađe bicikla, sklopa **PIO** spojenog na zvučnik (način postavljanja bitova) i sklopa **CT** (prekidni način, INT3). Na priključak CNT sklopa CT spojen je signal CLOCK frekvencije 10 kHz. CT nema priključak IACK.

Senzor vibracija postaje spreman:

a) ako nije bilo vibracija pa su počele; vrijednost podatka pročitanog sa senzora bit će 1
b) ako je bilo vibracija pa su prestale; vrijednost podatka pročitanog sa senzora bit će 0

Zvučnik može raditi na sljedeće načine:

a) **tiši** alarm: postavljanje jedinica na **nižih 4 bita** sklopa PIO
b) **glasniji** alarm: postavljanje jedinica na **svih 8 bitova** sklopa PIO
c) alarm **isključen**: postavljanje nula na **svih 8 bitova** sklopa PIO

Napišite program koji upravlja alarmom na sljedeći način:
Glavni program kontinuirano ispituje stanje senzora. Kada počnu vibracije, treba uključiti **tiši** alarm. Tada se **pokreće i CT** koji odbrojava **3 sekunde od uključivanja tišeg** alarma. Ako nakon **3 sekunde** vibracije nisu prestale, treba uključiti **glasniji** alarm. Ako **vibracije prestanu** (u bilo kojem trenutku), treba isključiti alarm.
Na početku rada, alarm treba biti isključen. Potprogram treba čuvati kontekst. Adrese vanjskih jedinica odabrati po volji.

RJEŠENJA

1. (1,5 bodova) Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **MOVE 50, R1**:

Razina dohvata:

Rastući brid CLOCK-a:

PC → AR

Padajući brid CLOCK-a:

PC+4 → PC

(AR) → IR, dekodiranje

ext 50 → ALU

Razina izvođenja:

Rastući brid CLOCK-a:

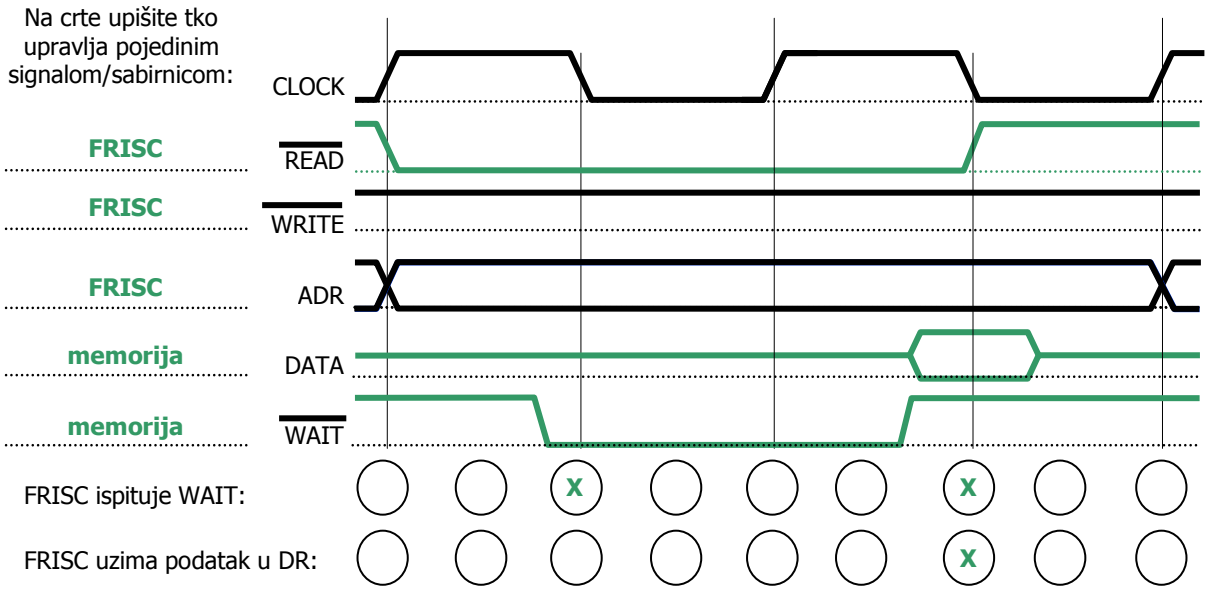
ALU: samo proslijeđuje drugi operand

Padajući brid CLOCK-a:

ALU → R1

2. (3,5 bodova) **Nacrtajte stanja na sabirnicama** FRISC-a za čitanje podatka iz memorije s jednim stanjem čekanja. Na crte s lijeve strane **upišite tko upravlja** pojedinom sabirnicom/signalom (FRISC ili memorija). **Stavite oznake** "X" u one kružiće u kojima:

- FRISC ispituje signal WAIT da utvrdi je li memorija obavila traženu operaciju (gornji red)
- FRISC čita podatak sa sabirnice podataka u interni registar DR (donji red)



3. (1,5 bodova) Odredite trajanje izvođenja sljedećeg programskog odsječka (pretpostavite da je memorija brza). Mora se vidjeti postupak računanja, tj. kraj svake naredbe napisati koliko puta se izvodi i koliko je njeno trajanje.

```
                `ORG
                JP GLAVNI              1 x 2 = 2
OSAM            `EQU 8
GLAVNI          MOVE OSAM, R0          1 x 1 = 1
PETLJA          ROTR R0, 2, R0         2 x 1 = 2
                JR_NC PETLJA           2 x 2 = 4
                STORE R0, (100)        1 x 2 = 2
                HALT                   1 x 2 = 2
```

Izvođenje traje 13 ciklusa.

4. (*6 bodova*) Na FRISC su spojeni **DMA** (adresa FFFF 1000), **bezuvjetna** vanjska jedinica BVJ (adresa FFFF 2000) i **uvjetne** vanjske jedinice UVJ1 i UVJ2 (adrese FFFF 3000 i FFFF 4000).

Napišite program koji treba prenijeti 100_{16} 32-bitnih podataka s BVJ u blok memorije od adrese 1000_{16}, pomoću **DMA prijenosa**. DMA treba raditi krađom ciklusa i pomoću prekida na INT0 dojaviti da je prijenos završen. Tijekom DMA prijenosa treba **prenositi podatke** iz UVJ1 na UVJ2 te u memorijskoj lokaciji BROJAC **prebrajati** koliko je podataka preneseno.

Nakon dovršenog DMA prijenosa, treba **prestati prenositi podatke** između uvjetnih vanjskih jedinica te početi izvoditi praznu petlju u glavnom programu. Potprogram treba čuvati kontekst.

```
DMASRC      `EQU  0FFFF1000
DMADEST     `EQU  0FFFF1004
DMACNT      `EQU  0FFFF1008
DMACTRL     `EQU  0FFFF100C
DMASTART    `EQU  0FFFF1010
DMAIACK     `EQU  0FFFF1014
BVJ         `EQU  0FFFF2000
UVJ1_DATA   `EQU  0FFFF3000
UVJ1_TEST   `EQU  0FFFF3004
UVJ2_DATA   `EQU  0FFFF4000
UVJ2_TEST   `EQU  0FFFF4004
      `ORG 0
      MOVE 10000, SP          ; inicijalizacija stoga
      JP GLAVNI               ; prijelaz u glavni program
      `ORG 8                  ; na adresi 8
      DW 500                  ; je adresa prekidnog vektora
GLAVNI
      MOVE %B 10010000, SR    ; omogućujemo prekid INT0
      MOVE BVJ, R0            ; izvor je BVJ
      STORE R0, (DMA_SRC)
      MOVE 1000, R0           ; odredište je adresa 1000
      STORE R0, (DMA_DEST)
      MOVE 100, R0            ; 100 podataka
      STORE R0, (DMA_CNT)
      MOVE %B 0111, R0        ; mem <- vj, krađa, prekid
      STORE R0, (DMA_CTRL)
      STORE R0, (DMA_START)
PETLJA
      LOAD R0, (GOTOVO)       ; je li u prekidu postavljen flag GOTOVO
      CMP R0, 1
      JR_EQ KRAJ

      LOAD R0, (UVJ1_TEST)    ; čekamo da UVJ1 postane spremna
      OR R0, R0, R0
      JR_Z PETLJA
      LOAD R0, (UVJ1_DATA)    ; uzimamo podatak s UVJ1
      STORE R0, (UVJ1_TEST)   ; brišemo spremnost UVJ1
PETLJA2
      LOAD R0, (UVJ2_TEST)    ; i da UVJ2 postane spreman
      OR R0, R0, R0
      JR_Z PETLJA2

      STORE R0, (UVJ2_DATA)   ; šaljemo podatak na UVJ2
      STORE R0, (UVJ2_TEST)   ; brišemo spremnost UVJ2

      LOAD R0, (BROJAC)       ; čitamo broj trenutno prenesenih
      ADD R0, 1, R0           ; povećavamo
      STORE R0, (BROJAC)      ; i spremamo broj natrag

      JR PETLJA               ; skok na početak petlje
```

```
KRAJ  JR KRAJ                  ; beskonačna petlja

BROJAC DW 0                    ; brojač prenesenih podataka
GOTOVO DW 0                    ; je li gotov DMA prijenos

      ;;;; prekidni potprogram
      `ORG 500                 ; adresa prekidnog potprograma
      PUSH R0                  ; čuvanje konteksta
      MOVE SR, R0
      PUSH R0

      STORE R0, (DMA_IACK)     ; prihvaćamo DMA prekid
      MOVE 1, R0               ; označavamo da treba prestati s UVJ1->UVJ2
      STORE R0, (GOTOVO)       ; i spremamo to u memoriju za glavni program

      POP R0                   ; obnavljanje konteksta
      MOVE R0, SR
      POP R0
      RETI                     ; povratak iz prekidnog potprograma
```

5. (7,5 *bodova*) Na FRISC je spojen alarm za bicikl koji se sastoji od **uvjetne vanjske jedinice** spojene na senzor vibracija – detektira pokušaj krađe bicikla, sklopa **PIO** spojenog na zvučnik (način postavljanja bitova) i sklopa **CT** (prekidni način, INT3). Na priključak CNT sklopa CT spojen je signal CLOCK frekvencije 10 kHz. CT nema priključak IACK.

Senzor vibracija postaje spreman:

a) ako nije bilo vibracija pa su počele; vrijednost podatka pročitanog sa senzora bit će 1

b) ako je bilo vibracija pa su prestale; vrijednost podatka pročitanog sa senzora bit će 0

Zvučnik može raditi na sljedeće načine:

a) **tiši** alarm: postavljanje jedinica na **nižih 4 bita** sklopa PIO

b) **glasniji** alarm: postavljanje jedinica na **svih 8 bitova** sklopa PIO

c) alarm **isključen**: postavljanje nula na **svih 8 bitova** sklopa PIO

Napišite program koji upravlja alarmom na sljedeći način:

Glavni program kontinuirano ispituje stanje senzora. Kada počnu vibracije, treba uključiti **tiši** alarm. Tada se **pokreće i CT** koji odbrojava **3 sekunde od uključivanja tišeg** alarma. Ako nakon **3 sekunde** vibracije nisu prestale, treba uključiti **glasniji** alarm. Ako **vibracije prestanu** (u bilo kojem trenutku), treba isključiti alarm.

Na početku rada, alarm treba biti isključen. Potprogram treba čuvati kontekst. Adrese vanjskih jedinica odabrati po volji.

Ideja rješenja je da se u GP provjerava spremnost uvjetnog senzora, sazna jesu li vibracije počele ili završile, i uključi/isključi (manji) alarm. Paralelno se uključuje CT na 3 sekunde. Ako tijekom 3 sekunde prestanu vibracije, isključuje se CT i alarm, i pri sljedećoj vibraciji ide se **ispočetka**, neovisno je li to sve bilo unutar 3 sekunde ili ne.

```
UVJ_DATA    `EQU 0FFFF0000
UVJ_TEST    `EQU 0FFFF0004
PIO_C       `EQU 0FFFF1000
PIO_D       `EQU 0FFFF1004
PIO_IACK    `EQU 0FFFF1008
PIO_IEND    `EQU 0FFFF100C
CT_LR       `EQU 0FFFF2000
CT_CR       `EQU 0FFFF2004
CT_IACK     `EQU 0FFFF2008
CT_IEND     `EQU 0FFFF200C
```

```
      `ORG 0
      MOVE 10000, SP
      JP GLAVNI

      `ORG 0C                 ; nemaskirajući prekid
      JP PP

GLAVNI
      MOVE %B 100, R0         ; ICR: postavlj. bitova, ne postavlja prekid
      STORE R0, (PIO_C)
      MOVE 0, R0              ; na početku je alarm isključen
      STORE R0, (PIO_D)

PETLJA
      LOAD R0, (UVJ_TEST)     ; čitanje spremnosti senzora
      OR R0, R0, R0
      JR_Z PETLJA

      LOAD R0, (UVJ_DATA)     ; čitanje podatka s UVJ
      STORE R0, (UVJ_TEST)    ; brisanje spremnosti
      CMP R0, 0               ; ako je stanje 0: prestanak vibracija
      JR_EQ OFF               ; ... idi na isključivanje

ON    MOVE %B 1111, R0        ; postavljanje tišeg alarma; niža 4 bita
      STORE R0, (PIO_D)
      MOVE %D 30000, R0       ; konstanta, 3 sekunde * 10 kHz
      STORE R0, (CT_LR)
      MOVE %B 11, R0          ; brojilo broji i postavlja prekid
      STORE R0, (CT_CR)
      JR PETLJA               ; povratak na ispitivanje spremnosti
OFF   MOVE 0, R0
      STORE R0, (CT_CR)       ; zaustavljamo brojilo
      STORE R0, (PIO_D)       ; isključujemo alarm

      JR PETLJA

      ;;;; prekidni potprogram
PP    PUSH R0                 ; spremamo kontekst
      MOVE SR, R0
      PUSH R0

      STORE R0, (CT_IACK)     ; prihvaćamo prekid

      MOVE %B 11111111, R0    ; uključujemo glasniji alarm
      STORE R0, (PIO_D)

      MOVE %B 00, R0          ; isključujemo brojilo
      STORE R0, (CT_CR)

      STORE R0, (CT_IEND)     ; kraj posluživanja

      POP R0                  ; obnavljamo kontekst
      MOVE SR, R0
      POP R0

      RETN                    ; povratak iz potprograma
```

Završni ispit iz Arhitekture računala 1 29.6.2006.

Prezime i ime (štampano): **MBR:**

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Potpis:_____

Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Rješenja obavezno pisati čitljivo, štampanim slovima. Koristite običnu olovku kako bi eventualne greške mogli popraviti.

Teoretski zadatci (ispunjavate na ovom papiru):

1. (0,5 boda) Zadan je binarni broj 1100. Ako je to zapis u 4-bitnom NBC-u, onda je to zapis broja _____. Ako je to zapis u 4-bitnom formatu dvojnog komplementa, onda je to zapis broja _____.

2. (0,5 boda) Ako smo zbrajali dva broja u NBC formatu i ako je došlo do prekoračenja opsega (tj. greške), onda je zastavica _____ u stanju _____.

3. (0,75 boda) Tri osnovna koraka koja procesori obavljaju prilikom izvođenja programa su: _____, _____ i _____

4. (1 bod) Za CISC i RISC arhitekturu nabrojeno je nekoliko svojstava. Za svako svojstvo označite sa "x" da li pripada jednoj ili drugoj arhitekturi:

Svojstvo	CISC	RISC
velik broj registara		
velik broj različitih naredaba		
složeno projektiranje arhitekture		
dulji programi		

5. (0,75 boda) Dopunite opis programskog brojila (PC). PC je _____ u kojem procesor pamti _____koju treba sljedeću izvesti. Nakon izvođenja svake naredbe, PC se automatski _____.

6. (0,25 boda) Povratna adresa iz potprograma kod FRISC-a se sprema (gdje?) _____.

7. (1,5 bod) Označite smjerove FRISC-ovih priključaka: READ je _____, WRITE je _____, ADR je _____, DATA je _____, WAIT je _____, INT2 je _____.

8. (1,75 boda) Za bezuvjetni, uvjetni i prekidni prijenos nabrojeno je nekoliko svojstava. Za svako svojstvo označite sa "x" čemu pripada:

Svojstvo	Bezuvjetni	Uvjetni	Prekidni
najbrži rad			
mogućnost gubitka/uvišestručenja podataka			
najsporiji rad			
vanjska jedinica ima bistabil stanja			
najjednostavnija vanjska jedinica			
pogodno za očitavanje temperature svake minute			

9. (0,25 boda) Ako imamo više neovisnih uvjetnih vanjskih jedinica, možemo primijeniti postupak koji se zove _____.

10. (0,5 boda) Kod sklopa FRISC-PIO maska se koristi u (kojem načinu rada?) _____, a maska zadaje _____.

11. (1,5 boda) FRISC-DMA ima u svojoj unutrašnjosti sljedeće dijelove: _____, _____, _____, _____ i _____.

Okreni!

12. (0,25 boda) Pojava kad procesor u određenom trenutku ne može izvesti sve faze onih naredaba koje se nalaze u protočnoj strukturi, jer sklopovlje procesora ne omogućuje istodobno izvođenje svih tih faza, naziva se _____.

13. (0,25 boda) Kad naredba koja se nalazi u protočnoj strukturi i spremna je za izvođenje nije naredba koja se u stvari treba izvesti, to se naziva _____.

14. (0,25 boda) Kada se kod izvođenja naredaba u protočnoj strukturi naredba ne može izvesti jer podaci potrebni za njeno izvođenje još nisu spremni, to se naziva _____.

15. (0,25 boda) Procesor ARM povratnu adresu iz potprograma sprema u _____.

16. (1 bod) Nakon uključenja procesor ARM treba izvesti program od sljedećih 6 naredaba. Koliko vremenskih perioda treba da se izvedu sve naredbe uključujući i zadnju naredbu ADD? Rješenje: _____

```
    'ORG 0

    MOV   R0,#2

A   SUBS  R0,R0,#1
    ADD    R2,R2,R2
    SUB     R2,R2,R2
    BNE A

    ADD    R0,R0,#1
```

17. (1,5 boda) Pretpostavimo da je u sustavu priručna memorija sa 32_{10} bloka. Podaci se u priručnu memoriju preslikavaju korištenjem algoritma za direktno preslikavanje (objašnjenom na predavanjima). Za zadane adrese blokova u radnoj memoriji odredite na koje će se adrese blokova u priručnoj memoriji preslikati blokovi iz radne memorije:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji (rješenja)
0x1	
0x2	
0x20	
0x152	
0x830	
0x44	

18. (0,5 boda) Za jednostavno statičko predviđanje grananja objašnjeno na predavanjima vrijedi sljedeće. Ako vrijedi da je _____, onda procesor pretpostavlja da je to skok na početak petlje pa kao sljedeću naredbu učitava naredbu zadanu adresom grananja.

19. (0,75 boda) Kod ARM-a, GPIO i RTC se spajaju na sabirnicu _____ (naziv?). Memorija i procesor ARM se spajaju sa sabirnicom _____ (naziv?). Između ovih dvaju sabirnica nalazi se sklop koji se zove _____.

Rješenja teoretskih zadataka

1. (0,5 boda) Zadan je binarni broj 1100. Ako je to zapis u 4-bitnom NBC-u, onda je to zapis broja **12**. Ako je to zapis u 4-bitnom formatu dvojnog komplementa, onda je to zapis broja **-4**.

2. (0,5 boda) Ako smo zbrajali dva broja u NBC formatu i ako je došlo do prekoračenja opsega (tj. greške), onda je zastavica **prijenos** u stanju **1**.

3. (0,75 boda) Tri osnovna koraka koja procesori obavljaju prilikom izvođenja programa su: **dohvat**, **dekodiranje** i **izvođenje**.

4. (1 bod) Za CISC i RISC arhitekturu nabrojeno je nekoliko svojstava. Za svako svojstvo označite sa "x" da li pripada jednoj ili drugoj arhitekturi:

Svojstvo	CISC	RISC
velik broj registara		X
velik broj različitih naredaba	X	
složeno projektiranje arhitekture	X	
dulji programi		X

5. (0,75 boda) Dopunite opis programskog brojila (PC). PC je **registar** u kojem procesor pamti **adresu naredbe** koju treba sljedeću izvesti. Nakon izvođenja svake naredbe, PC se automatski **povećava**.

6. (0,25 boda) Povratna adresa iz potprograma kod FRISC-a se sprema (gdje?) **na stog**.

7. (1,5 bod) Označite smjerove FRISC-ovih priključaka: READ je **izlazni**, WRITE je **izlazni**, ADR je **izlazni**, DATA je **dvosmjerni**, WAIT je **ulazni**, INT2 je **ulazni**.

8. (1,75 boda) Za bezuvjetni, uvjetni i prekidni prijenos nabrojeno je nekoliko svojstava. Za svako svojstvo označite sa "x" čemu pripada:

Svojstvo	Bezuvjetni	Uvjetni	Prekidni
najbrži rad	X		
mogućnost gubitka/uvišestručenja podataka	X		
najsporiji rad		X	
vanjska jedinica ima bistabil stanja		X	X
najjednostavnija vanjska jedinica	X		
pogodno za očitavanje temperature svake minute	X		

9. (0,25 boda) Ako imamo više neovisnih uvjetnih vanjskih jedinica, možemo primijeniti postupak koji se zove **prozivanje**.

10. (0,5 boda) Kod sklopa FRISC-PIO maska se koristi u (kojem načinu rada?) **načinu ispitivanja bitova**, a maska zadaje **koji bitovi se ispituju**.

11. (1,5 boda) FRISC-DMA ima u svojoj unutrašnjosti sljedeće dijelove: **upravljački dio**, **upravljački registar**, **međuspremnik**, **brojilo podataka**, **adresni registar izvora** i **adresni registar odredišta**.

12. (0,25 boda) Pojava kad procesor u određenom trenutku ne može izvesti sve faze onih naredaba koje se nalaze u protočnoj strukturi, jer sklopovlje procesora ne omogućuje istodobno izvođenje svih tih faza, naziva se **strukturni hazard**.

13. (0,25 boda) Kad naredba koja se nalazi u protočnoj strukturi i spremna je za izvođenje nije naredba koja se u stvari treba izvesti, to se naziva **upravljački hazard**.

14. (0,25 boda) Kada se kod izvođenja naredaba u protočnoj strukturi naredba ne može izvesti jer podaci potrebni za njeno izvođenje još nisu spremni, to se naziva **podatkovni hazard**.

15. (0,25 boda) Procesor ARM povratnu adresu iz potprograma sprema u **registar R14 (ili LR)** .

16. (1 bod) Nakon uključenja procesor ARM treba izvesti program od sljedećih 6 naredaba. Koliko vremenskih perioda treba da se izvedu sve naredbe uključujući i zadnju naredbu ADD? Rješenje: **14**

```
    'ORG 0

    MOV   R0,#2

A   SUBS  R0,R0,#1
    ADD    R2,R2,R2
    SUB     R2,R2,R2
    BNE A

    ADD    R0,R0,#1
```

17. (1,5 boda) Pretpostavimo da je u sustavu priručna memorija sa 32_{10} bloka. Podaci se u priručnu memoriju preslikavaju korištenjem algoritma za direktno preslikavanje (objašnjenom na predavanjima). Za zadane adrese blokova u radnoj memoriji odredite na koje će se adrese blokova u priručnoj memoriji preslikati blokovi iz radne memorije:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji (rješenja)
0x1	**0x1**
0x2	**0x2**
0x20	**0x0**
0x152	**0x12**
0x830	**0x10**
0x44	**0x4**

18. (0,5 boda) Za jednostavno statičko predviđanje grananja objašnjeno na predavanjima vrijedi sljedeće. Ako vrijedi da je **adresa grananja manja od PC-a**, onda procesor pretpostavlja da je to skok na početak petlje pa kao sljedeću naredbu učitava naredbu zadanu adresom grananja.

19. (0,75 boda) Kod ARM-a, GPIO i RTC se spajaju na sabirnicu **APB** (naziv?). Memorija i procesor ARM se spajaju sa sabirnicom **AHB** (naziv?). Između ovih dvaju sabirnica nalazi se sklop koji se zove **most**.

Programski zadatci (rješavate na svojim papirima):

20. (2 boda) Za **FRISC** napisati potprogram koji predznačno proširuje 16-bitni broj na 32 bita. 16-bitni broj zapisan je u memoriji, a njegova adresa se prenosi stogom kao parametar potprograma (parametar uklanja pozivatelj). Rezultat se vraća preko R0. **Potprogram <u>mora</u> čuvati registre.**

```
SIGN_EXTEND
    PUSH R1                  ; kontekst (push-pop)

    LOAD R1, (SP+8)          ; učitavanje adrese sa stoga
    LOADH R0, (R1)           ; učitavanje podatka sa adrese

    ROTL R0, 10, R0          ; zajedno sa ROTR, ispitivanje predznaka
    JR_P POZIT               ; grananje
NEGAT
    OR R0, 0FFFF, R0         ; proširivanje
POZIT
    ROTR R0, 10, R0

    POP R1
    RET                      ; povratak
```

```
Napomena: 2006. je korištena starija verzija FRISC-a koja je imala samo naredbe ROTR
i ROTL, a  sadašnja verzija ima još i SHR, SHL i ASHR. Tako da se proširivanje može
napraviti i jednostavnije:

    ROTL R0, 10, R0
    ASR  R0, 10, R0
```

21. (5 bodova) Za **ARM** napisati potprogram koji računa sumu N članova sljedećeg niza: Y(X,N) = X*2 + X*4 + X*8 + X*16... + X*2^N. Parametri X i N prenose se u potprogram preko registara R0=X i R3=N, a suma Y se vraća preko registra R0. Napisati i glavni program koji poziva potprogram s parametrima X=4 i N=5. Potprogram **<u>ne mora</u>** čuvati registre. Zanemariti slučajeve prekoračenja opsega.

```
      `ORG 0                 ;glavni program
      MOV R0, #4             ; učitavanje parametara
      MOV R3, #5
      BL NIZ1                ; poziv
      SWI 123456             ; završetak

      ; potprogram
NIZ1  MOV R1,#1              ; inicijalizacija brojača
      MOV R2,#0              ; SUMA inicijalizacija

NIZ1_PETLJA
      ADD R2,R2,R0, LSL R1   ; množi s potencijom (LSR brojač) i sumira
      ADD R1,R1,#1           ; povećanje brojača
      CMP R1,R3              ; da li je N članova
      BLS NIZ1_PETLJA        ; skok

NIZ1_VAN
      MOV R0,R2              ; rezultat u R0
      MOV PC, LR             ; povratak iz potp.
```

22. (9 bodova) Na procesor **ARM** spojeni su RTC (adresa 0FFFFFF00) i GPIO (adresa 0FFFFFE00). Napisati program koji korištenjem prekida IRQ, sa sklopa RTC, svake sekunde mijenja stanje na bitu 5 (početno stanje je 0) od vrata A sklopa GPIO. Na ulaz RTC-a doveden je signal frekvencije 10 Hz. **Prekidni potprogram <u>mora</u> čuvati sve registre**, a glavni program vrti beskonačnu petlju.

```
      `ORG 0
      B GLAVNI          ; skok "preko prekidnog potprograma" u glavni


      `ORG 18
PREKIDNI
      STMFD R13!, {R0, R2, R3}      ; spremi kontekst + obnovi kontekst

      LDR R2,LGPIO      ; učitavanje adrese RTC-a i GPIO-a
      LDR R3,LRTC

      LDR R0,[R2]       ; promjena bita 5
      EOR R0,R0,#20
      STR R0,[R2]

      STR R0,[R3,#8]    ; obrađen prekid RTC-a
      MOV R0,#0
      STR R0,[R3,#0C]   ; vrati brojilo na nulu

      LDMFD R13!, {R0, R2, R3}      ; obnovi kontekst
      SUBS PC,R14,#4    ; povratak


GLAVNI
      MOV R13,#1<8
      LDR R2,LGPIO      ; GPIO - učitavanje adrese
      LDR R3,LRTC       ; RTC - učitavanje adrese

      MOV R0,#A         ; 1 sekunda (10 dekadski)
      STR R0,[R3,#4]    ; upis

      MOV R0,#1         ; RTC generira prekid
      STR R0,[R3,#10]   ; upis

      MRS R0,CPSR       ; omogući prihvaćanje prekida
      BIC R0,R0,#80
      MSR CPSR_c,R0

      MOV R0, #20       ; postavljanje bita 5 da je izlazni
      STR R0, [R2,#8]   ; upis

      MOV R0,#0         ; početno stanje bita je 0 - nije nužno, jer 0 je default
      STR R0, [R2]      ; upis

PETLJA                  ; beskonačna petlja
      B PETLJA


      `ORG 100
LGPIO DW 0FFFFFE00      ; adrese sklopova RTC i GPIO
LRTC  DW 0FFFFFF00
```

Završni ispit iz Arhitekture računala 1 — 21. lipnja 2007.

Prezime i ime (štampano):_____ **MBR:** _____

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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće.

Potpis:_____

PAŽLJIVO PROČITATI: Dozvoljeno je koristiti isključivo službene šalabahtere (popis naredaba FRISC-a i ARM-a). Rješenja obavezno pisati čitljivo, štampanim slovima. Koristite običnu olovku kako bi eventualne greške mogli popraviti. Programe treba pisati uredno i komentirati pojedine cjeline programa. Osim u 1 i) zadatku, nema negativnih bodova.

Teoretski zadatci (ispunjavate na ovom papiru):

1 a) (0,5 bod) Zadan je podatak -3. Njegov prikaz u 5-bitnom formatu 2'k izgleda ovako: **11101** (prikažite binarno), a prikaz u 6-bitnom formatu s bitom za predznak izgleda ovako **100011** (prikažite binarno).

1 b) (0,75 boda) Kod arhitekture registar-registar (load-store), u ALU-naredbama se prvi operand nalazi u **registru**, drugi operand u **registru**, a rezultat se sprema u **registru**.

1 c) (0,25 boda) Troprolazni asembleri, za razliku od dvoprolaznih asemblera, mogu prevoditi programe koji koriste **makronaredbe**.

1 d) (0,5 boda) U stogovnom okviru FRISC-a nalaze se povratna adresa (A), parametri (B), spremljeni registri (C). Počevši od viših adresa poredajte ove tri vrste podataka (A, B i C): **B**, **A**, i na najnižoj adresi **C**.

1 e) (1,5 boda) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: **ADR**, **DATA**, **READ**, **WRITE**, **WAIT** i **SIZE**.

1 f) (1,25 bod) FRISC-ov sklop PIO može raditi u sljedeća četiri načina rada: **ulazni način**, **ispitivanje bitova**, **izlazni način** i **postavljanje bitova**. Maska se koristi samo u jednom od tih načina rada i to u **ispitivanju bitova**.

1 g) (1 bod) DMA može obavljati prijenos na četiri načina. Njihovi nazivi su: **krađa ciklusa**, **zaustavljanje procesora**, **blokovski prijenos** i **multipleksirani prijenos**.

1 h) (1,5 bod) Dopunite sljedeće korake kod prihvaćanja **IRQ** na ARM-u:

R14_irq ← R15 // upisati ime registra
SPSR_irq ← CPSR // upisati ime registra
CPSR [4:0] ← način rada IRQ // ne upisivati broj, već što taj broj znači, kao u ovom primjeru
CPSR [5] ← **način rada ARM** // bit T
CPSR [6] ← **ne mijenja se** // bit F
CPSR [7] ← **zabrani IRQ** // bit I
PC ← **18 (heksa)** // upisati adresu

1 i) (1 bod) Za ARM-ove vanjske jedinice zaokružite točne odgovore. Točan odgovor nosi 0,2, a netočan -0,2 boda.
Ima li GPIO ima sinkronizacijske priključke? da **ne**
Može li GPIO postaviti zahtjev za prekid? da **ne**
Koliko se u GPIO-u bitova koristi u registrima smjera GPIODDR? 1 bit **8 bita**
Može li RTC nakon odbrojavanja jednog ciklusa automatski nastaviti s brojenjem sljedećeg ciklusa? da **ne**
Može li RTC postaviti zahtjev za prekid? **da** ne

Okreni!

1 j) (1 bod) Koliko ukupno perioda traje izvođenje pojedinih naredaba na arhitekturi ARM 7, tj. koliko perioda se izvodi naredba ne računajući preklapanje u protočnoj strukturi? LDR traje _**5**___ perioda. BL traje _**5**__ perioda. ADD traje _**3**__ perioda. ADDEQ traje __**3**_ perioda.

1 k) (0,25 boda) Podatkovni hazard može se javiti na arhitekturi ARM 9. Dopunite naredbu tako da dođe do podatkovnog hazarda:

```
ADD  R1, R2,  R3
SUB  R5, _R1_, R7
```

1 l) (0,5 boda) Kod statičkog predviđanja grananja se predviđanje ostvaruje usporedbom dvaju podataka (tj. adresa). Zapravo se uspoređuju __**programsko brojilo (ili PC ili R15)**_ i _____**adresa skoka (ili odredište skoka)**_____.

1 m) (1 bod) Nakon uključenja procesor ARM treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:__**11**__

```
      `ORG 0
      MOV R0, #2
LAB   SUBS  R0,R0,#1
      BNE LAB
      STR R0,[R1]
```

1 n) (0,5 boda) Vrijeme pristupa memorije obično se izračunava za čitanje podatka. Pri tome se gleda razdoblje od trenutka kada __ **procesor postavi upravljačke signale i adresu (ili započne s pristupom)** _____ pa do trenutka kada ____**memorija obavi traženu operaciju (ili memorija obavi čitanje/pisanje)**_____.

1 o) (0,5 boda) U sustavu je memorija s 8-strukim preplitanjem. Neka postavljanje adrese i upravljačkih signala traje 1 takt, vrijeme pristupa je 6 taktova, a čitanje/pisanje podatka traje 1 takt. Izračunajte koliko taktova će trajati pristup do 5 slijednih podataka. Pristup će trajati _**12**__ taktova.

1 p) (1 bod) Neka priručna memorija ima 16_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
0x2	**0x2**
0x21	**0x1**
0x22	**0x2**
0x153	**0x3**

Programski zadatci (rješavate na svojim papirima):

2. (4 boda) Na FRISC su spojene uvjetna vanjska jedinica UVJ na adresi $FFFF0000_{16}$ i bezuvjetna BVJ na adresi $FFFF8880_{16}$. Program treba prenositi podatke sa UVJ na BVJ. Podatak se prenosi ako je paran, a neparni podatci se zanemaruju. Nakon što se na BVJ pošalje 1000_{10} podataka, treba zaustaviti procesor.

```
BVJ    `EQU FFF8880

VJUDATA      `EQU FFFF0000
VJUSTAT      `EQU FFFF0004
VJUCLEAR     `EQU FFFF0004          //definiranje adresa

MAIN  MOVE %d 1000, R2              //init brojača

LOOP  LOAD R0, (VJUSTAT)            //ispitivanje spremnosti
      CMP R0, 1
      JP_NE LOOP

      LOAD R0, (VJUDATA)            //čitanje podataka i brisanje spremnosti
      STORE R0, (VJUCLEAR)

      ROTR R0, 1, R1                //ispitivanje parnosti
      JP_C LOOP

PARAN STORE R0, (BVJ)               //slanje na vanjsku i brojanje
      SUB R2, 1, R2
      JP_NZ LOOP                    //zaustavljanje
      HALT
```

3. (6 bodova) U memoriji procesora ARM nalazi se popis od 512_{10} 32-bitnih podataka o studentima koji su dolazili na predavanja. Zapisani podatak predstavlja identifikacijski broj studenta *ID* (broj između 0 i 100_{16}) koji je došao na predavanje. Pretpostavka je da se evidentira samo ulaz na predavanje, a izlaz se ne bilježi.

Napisati potprogram *BROJI* koji će prebrojati na koliko je predavanja student prisustvovao i napraviti listu prisutnosti za sve studente. Potprogram preko stoga prima dva ulazna parametra: adresu popisa i adresu izlazne liste prisutnosti. Potprogram zapisuje u listu 8-bitne podatke o prisutnosti (dakle na početnoj adresi liste je podatak o broju prisutnosti predavanjima za studenta s *ID* brojem 0, na sljedećoj adresi podatak za studenta s *ID* brojem 1, itd.). Dodatno, potprogram na kraju treba pronaći identifikacijski broj studenta koji je bio na najviše predavanja (pretpostaviti da postoji samo jedan takav) i njegov *ID* vratiti preko registra *R0* iz potprograma. Potprogram treba čuvati stanja registara.

```
BROJI      STMFD R13!, {R1,R2,R3,R4,R5} //spremanje konteksta

           ADD R5,R13,#14        // pomak da se ucitaju podaci sa stoga
           LDMFD R5, {R0,R1}     // ucitaj parametre
                                 // R0 je popis, R1 je lista

           MOV R2, #2<8          // brojac 512 podataka

POD1       LDR R3, [R0], #4      // ucitaj prvi podatak
           LDRB R4, [R1,R3]      // ucitaj dosadasnji broj ulaza
           ADD R4,R4,#1          // povecaj za 1
           STRB R4, [R1,R3]      // spremi uvecani broj ulaza
           SUBS R2,R2,#1         // vrti petlju 512 puta
           BNE POD1              // petlja

           //pronalazenje tko je bio najvise puta

           MOV R2, #1            // brojac od 1 do 256
           MOV R5, #0            // ID najveceg na pocetku
           LDR R3, [R1]          // ucitavanje prvog koji je pocetno najveci

P2         LDR R4, [R1,R2]       // ucitavaje slijedeceg
           CMP R4,R3             // usporedba
           MOVHI R3,R4           // (može i MOVGT) nova vrijednost najveceg
           MOVHI R5,R2           // (može i MOVGT) novi ID najveceg
           ADD R2,R2,#1          // povecavanje brojaca
           CMP R2,#1<8           // i to ponovi 256 puta
           BLS P2                // petlja

           MOV R0, R5            // povratna vrijednost

           LDMFD R13!, {R1,R2,R3,R4,R5} //obnova konteksta

           MOV PC,R14            // povratak
```

4. (7 bodova) U sustavu s procesorom ARM nalaze se RTC (na adresi $FFFF1000_{16}$) i GPIO (na adresi $FFFF2000_{16}$). RTC broji vanjske impulse i nakon svakih 1000_{16} impulsa generira iznimku FIQ. U prekidnom potprogramu treba pročitati vrijednost postavljena na vrata B GPIO sklopa, te izračunati paritet pročitanog podatka. Ako pročitani podatak ima paran paritet tada se šalje na vrata A GPIO sklopa, a u slučaju neparnog pariteta ne radi se ništa. Funkciju računanja pariteta riješiti potprogramom *PAR*. Nakon što se primi 300_{16} podataka, treba onemogućiti daljnje generiranje iznimaka od strane RTC-a. Potprogram treba čuvati stanja registara.

```
            `ORG 0
            B GLAVNI                // skok u glavni

            `ORG 1C                         // adresa iznimke FIQ
            STMFD R13!,{R0,R1,R2,R3,R4,R14} // spremanje konteksta
            LDR R3, GPIO
            LDR R4, RTC
            LDR R0, [R3,#GPIOPBDR]          // citaj podatak sa B
            BL PAR                          // poziv potrpograma
            CMP R2, #0
            STREQ R0, [R3,#GPIOPADR]        // ako je paran, šalji na A

            MOV R0, #0
            STR R0, [R4,#RTCCLR]            // resetira brojac
            STR R0, [R4,#RTCEOI]            // potvrduje obradu iznimke

            LDR R1, BROJAC        // smanjivanje brojaca podataka
            SUBS R1, R1, #1
            STR R1, BROJAC

            BNE VAN                         // ispitivanje da li je 300 podataka
            STR R0, [R4,#RTCCR]             // onemoguci daljnje iznimke
VAN         LDMFD R13!,{R0,R1,R2,R3,R4,R14} // obnova konteksta
            SUBS PC, LR, #4                 // povratak iz iznimke


BROJAC      DW 300              // definicija brojaca

GPIO        DW   0FFFF1000
GPIOADR     `EQU 0
GPIOBDR     `EQU 4
GPIOADDR    `EQU 8
GPIOBDDR    `EQU 0C

RTC         DW   0FFFF2000
RTCDR       `EQU 0
RTCMR       `EQU 4
RTCEOI      `EQU 8
RTCCLR      `EQU 0C
RTCCR       `EQU 10


// potprogram za računanje pariteta, proizvoljno: parametar=R0, rezultat=R2
PAR         STMFD R13!, {R1}      // spremanje konteksta

            MOV R1, #8            // prebrajanje jedinica za paritet
            MOV R2, #0            // brojač jedinica u podatku
PETLJA      MOVS R0, R0, LSR #1   // najniži bit u zastavicu C
            ADC R2, R2, #0        // (ili ADDCS R2,R2,#1) prebrajanje jedinica
            SUBS R1, R1, #1
            BNE PETLJA

            AND R2, R2, #1  // paritet paran => R2=0, paritet neparan => R2=1

            LDMFD R13!, {R1}      // obnova konteksta
            MOV PC, LR            // povratak iz potprograma
```

```
GLAVNI     MVN R0, #0
           LDR R3, GPIO
           LDR R4, RTC

           STR R0, [R3,#GPIOPADDR]   // port A izlazni
           STR R0, [R3,#GPIOPBDDR]   // port B ulazni

           MOV R0, #0
           STR R0, [R4,#RTCLR]       // init brojaca
           MOV R0, #1<12
           STR R0, [R4,#RTCMR]       // MR=1000
           MOV R0, #1
           STR R0, [R4,#RTCCR]       // omoguci prekid u RTC-u

           MRS R0, CPSR
           BIC R0, R0, #40     // omoguci prihvat FIQ
           MSR CPSR_c, R0

LOOP       B LOOP             // petlja cekalica
```

Komentar: u prekidnom potprogramu su se mogli koristiti registri r7, r8, ... pa bi onda bilo manje konteksta za spremati

Završni ispit iz Arhitekture računala 1 — 26. lipnja 2008.

Prezime i ime (štampanim 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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

Dozvoljeno je koristiti isključivo službene šalabahtere (popisi naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Ispit se piše 120 minuta.

Teoretski zadatci (ispunjavate na ovom papiru):

1. (*0,5 boda*) Zadan je binarni broj 1011_2. Ako je to zapis u 4-bitnom NBC-u, onda je to zapis broja ___**11**____. Ako je to zapis u 4-bitnom formatu dvojnog komplementa, onda je to zapis broja ____**-5**_____.

2. (*0,5 boda*) 01011_2 je 5-bitni prikaz u formatu dvojnog komplementa. Prikažite taj broj u 6-bitnom formatu s bitom za predznak_____**001011**_____. 11011_2 je prikaz u 5-bitnom formatu dvojnog komplementa. Prikažite taj broj u 4-bitnom formatu jediničnog komplementa_____**1010**_____.

3. (*0,25 boda*) Pojava kad procesor u određenom trenutku ne može izvesti sve faze onih naredaba koje se nalaze u protočnoj strukturi, jer sklopovlje procesora ne omogućuje istodobno izvođenje svih tih faza, naziva se _____**strukturni hazard**_____.

4. (*1 bod*) Neki procesor ima SP koji pokazuje na **prvo slobodno mjesto** na stogu, a **stog raste prema višim adresama**. Procesor adresira bajtove, a prilikom operacija PUSH i POP čita i piše 32-bitne riječi u formatu *little-endian*. Početno stanje registra SP je 100, a u memorijskim lokacijama su ništice. Na desnoj slici upišite vrijednosti u SP i memorijske lokacije nakon izvođenja naredaba:

PUSH 12345678_{16}
PUSH $99AABBCC_{16}$
POP

FB	itd.
FC	0
FD	0
FE	0
FF	0
100	78
101	56
102	34
103	12
104	CC
105	BB
106	AA
107	99
108	0
109	itd.

SP: 104

5. (*1,25 boda*) Rad procesora odvija se u tri osnovna koraka koji se stalno ponavljaju. To su: ____**dohvat**____, ____**dekodiranje**____, _____**izvođenje**_____. Od ova tri koraka, procesor sigurno ne pristupa memoriji u koraku _____**dekodiranje**_____, u koraku ___**dohvata**_____ sigurno pristupa memoriji, a u koraku _____**izvođenja**_____ može i ne mora pristupati memoriji.

6. (*0,25 boda*) Neki procesor ima 16 registara opće namjene i 18 različitih ALU naredaba i sve one imaju tri operanda koji mogu biti isključivo registri opće namjene. Za kodiranje ALU naredaba, strojni kod mora biti širok barem _**17**__ bitova.

7. (*2 boda*) Za pitanja o priključcima i sabirnicama procesora FRISC s lijeve strane, označite znakom "x" točan odgovor u pojedinim sivim kućicama (može biti više točnih odgovora).

Priključak READ je:	ulazni	izlazni	dvosmjerni
Adresni priključci su:	ulazni	izlazni	dvosmjerni
Podatkovni priključci su:	ulazni	izlazni	dvosmjerni
Adresne priključke FRISC postavlja u *high Z* onda kada	čita podatak	piše podatak	se obavlja DMA prijenos
Priključak IACK se koristi kod:	maskirajućih prekida	nemaskirajućih prekida	svih prekida
Sabirnica kod FRISC-a je:	sinkrona	asinkrona	niti jedno od ponuđenog
Sabirnica kod FRISC-a je:	multipleksirana	nemultipleksirana	niti jedno od ponuđenog
Koji od ovih priključaka su nužni za komunikaciju s memorijom:	SIZE	READ	ADR

8. (*1 bod*) Kada u sklopu FRISC-CT brojilo odbroji zadani broj ciklusa, događa se sljedeće (označite točne odgovore znakom "x" u pojedinim sivim kućicama).

brojilo se automatski inicijalizira na 0	u brojilo treba programski upisati 0	u brojilo se automatski upiše vrijednost iz LR	brojilo automatski nastavlja s brojenjem	brojilo prestane brojiti i treba ga programski ponovno pokrenuti	registar LR se automatski inicijalizira na 0
CT generira impuls na priključku ZC	CT generira impuls na priključku CNT	CT će sigurno generirati zahtjev za prekid	CT može generirati zahtjev za prekid	CT postaje spreman ako je prethodno bio poslužen do kraja	CT sigurno postaje spreman

9. (*0,5 boda*) Priključci READY i STROBE nazivaju se priključcima za ____**sinkonizaciju (rukovanje, handshaking)**_____, a od FRISC-ovih vanjskih jedinica ima ih sklop _____**PIO**_____.

10. (*0,25 boda*) Procesor ARM povratnu adresu iz potprograma sprema u ___**LR (R14)**_.

11. (*1 bod*) Nakon uključenja procesor ARM treba izvesti program od sljedećih 6 naredaba. Koliko vremenskih perioda treba da se izvedu sve naredbe uključujući i zadnju naredbu ADD? Rješenje: ___**12**_____.

```
'ORG 0
 MOV  R0,#2
A ADDNE R2,R2,R2
 SUBS  R0,R0,#1
 BNE   A
 ADD   R0,R0,#1
```

12. (*0,25 boda*) Procesor ARM izvršava programski odsječak s desne strane. Navedeni odsječak ima funkciju (zaokružite točan odgovor; ukoliko krivo zaokružite: *-0,1 bod*):
a) desnog posmaka donjih 8 bita registra R0 za 4 mjesta lijevo
b) oduzimanja registra R0 sa 4
c) oduzimanja registra R0 sa 16
d) množenja registra R0 sa 15

```
...
RSB R0, R0, R0 LSL #4
...
```

13. (*0,75 boda*) Kod ARM-a, GPIO i RTC se spajaju na sabirnicu ___**APB**_____. Memorija i procesor ARM se spajaju sa sabirnicom _____**AHB**_____. Između ovih dvaju sabirnica nalazi se sklop koji se zove____**most**_____ .

14. (*0,5 boda*) U sustavu procesora ARM signali *HADDR* su dio sabirnice ___**AHB**___ i širine su __**32**__ bita.

15. (*0,5 boda*) Nabrojite dvije naredbe za pristup registrima stanja procesora ARM: __**MRS**__, __**MSR**___.

16. (*0,75 boda*) Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak ____**88776655**___(0,5 boda)_____, a u registru R1 podatak __**104**____(0,25 boda)_____ .

```
`ORG 0
MOV  R1,#1<8
LDR  R0,[R1,#4]!
`ORG  100
DB 11, 22, 33, 44, 55, 66, 77, 88, 99, AA
```

17. (*0,75 boda*) Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak ____**11** (0,5 boda)_____, a u registru R1 podatak ____**104**___(0,25 boda)_____ .

```
`ORG 0
MOV  R1,#1<8
LDRSB  R0,[R1],#4
`ORG  100
DB 11, 22, 33, 44, 55, 66, 77, 88, 99, AA
```

Programski zadatci (rješavate na svojim papirima):

18. (*4 boda*) Na procesor FRISC spojene su dvije nezavisne uvjetne vanjske jedinice VJ1 i VJ2, te dvije bezuvjetne vanjske jedinice VJ3 i VJ4. Glavni program treba prenositi podatke sa VJ1 na VJ3, a podatke sa VJ2 treba prenositi na VJ4. Nakon što je preneseno ukupno 1000_{10} podataka treba zaustaviti procesor. Adrese jedinica odaberite po volji.

```
VJ1PRIMI        `EQU FFFF1000
VJ1STANJE       `EQU FFFF1004
VJ2PRIMI        `EQU FFFF2000
VJ2STANJE       `EQU FFFF2004
VJ3SALJI        `EQU FFFF3000
VJ4SALJI        `EQU FFFF4000
        MOVE 0, R0              ;BROJAC         ; adrese i inicijalizacija brojaca

PRVA    LOAD R1, (VJ1STANJE)                    ; ispitivanje spremnosti prve
        OR R1, R1, R1
        JP_Z DRUGA                              ; prozivanje(1)
        LOAD R1, (VJ1PRIMI)
        STORE R1, (VJ3SALJI)                    ; citanje i pisanje podataka s VJ-a
        STORE R1, (VJ1STANJE)
        ADD R0, 1, R0
        CMP R0, %D1000
        JP_EQ KRAJ                              ; uvecavanje brojaca, usporedba i skok
DRUGA   LOAD R1, (VJ2STANJE)                    ; ispitivanje spremnosti druge
        OR R1, R1, R1
        JP_Z PRVA                               ; prozivanje (2)
        LOAD R1, (VJ2PRIMI)
        STORE R1, (VJ4SALJI)                    ; citanje i pisanje podataka s VJ-a
        STORE R1, (VJ2STANJE)
        ADD R0, 1, R0
        CMP R0, %D1000
        JP_EQ KRAJ                              ; uvecavanje brojaca, usporedba i skok
        JP PRVA
KRAJ    HALT                                    ; prozivanje i kraj(3)
```

19. (4 *boda*) Za procesor ARM napisati potprogram PROSIRI koji predznačno proširuje 16-bitni 2'k broj na 32 bita. 16-bitni broj zapisan je u memoriji, a njegova adresa se prenosi stogom kao parametar potprograma (parametar uklanja pozivatelj). Rezultat se vraća preko R0.

Dodatno, napisati potprogram BLOK koji proširuje deset 16-bitnih 2'k podataka iz bloka od adrese 1000_{16}. Proširivanje se obavlja pozivom potprograma PROSIRI. Prošireni podaci se spremaju od adrese 2000_{16}.

```
PROSIRI
    LDR R0, [R13]             ; čita adresu sa stoga

    LDRSH R0, [R0]            ; proširivanje
ili ***
    LDRH R0, [R0]
    MOV R0, R0, LSL #%D 16
    MOV R0, R0, ASR #%D 16
    ***

    MOV PC, LR                ; povratak


BLOK    STMFD R13!, {R1, R2, R3, LR}      ;kontekst

    MOV R3, #%D 10                        ; R5 - brojac 10
    LDR R1, SRC                           ; ili MOV R1, #1<12; R1 - source addr
    LDR R2, DEST                          ; ili MOV R2, #2<12; R2 - dest addr

PETLJA    STMFD R13!, {R1}                ; R1 na stog (adresa podatka)
    BL PROSIRI
    LDMFD R13!, {R1}                      ; pozivatelj mora ukloniti R1

    STR R0, [R2], #4    ; stavi prosireni podatak u memoriju i povecaj dest adresu
    ADD R1, R1, #2                        ; povecaj source adresu

    SUBS R3, R3, #1                       ; smanji brojac
    BNE PETLJA

    LDMFD R13!, {R1, R2, R3, LR}          ; kontekst
    MOV PC, LR                            ; povratak

SRC    DW 1000
DEST   DW 2000
```

20. (*10 bodova*) RTC sklopom na ARM-u generira se brzi prekid (FIQ) svake 2 sekunde. Na RTC je spojen signal frekvencije 10kHz.

U prekidnom potprogramu za obradu prekida RTC-a pročitajte 8 bitni podatak u NBC formatu sa vrata B GPIO sklopa. Pročitani podatak podijelite sa 4 korištenjem potprograma DIJELI i pošaljite ga na vrata A GPIO sklopa.

Napišite potprogram DIJELI koji prima 8 bitni podatak u NBC formatu preko registra R0, dijeli ga sa 4 i rezultat (isto u NBC formatu) vraća u registru R0.

Sve potrebne inicijalizacije RTC i GPIO sklopova obavite u glavnom programu. Adrese sklopova odaberite po volji. Nakon inicijalizacije glavni program beskonačno izvodi praznu petlju. Potprogrami moraju čuvati stanja registara koje koriste za obradu podataka.

Podsjetnik: **registar CPSR**: bit 7 zastavica *I*, bit 6 zastavica *F*, bitovi 0-4 *način rada*.

```
    'ORG 0                            ; boda
    B GLAVNI

    'ORG 1C                           ; adresa
    ; R8 - R12 su raspolozivi njih ne treba spremati
    STMFD R13!,{R0,R1,R14}            ; kontekst

    MOV R0,#1<8                       ; adrese:
```

```
      LDR R9,[R0],#4                    ; GPIO
      LDR R10,[R0]                      ; RTC
      MOV R11,#0                        ; za reset RTC-a u FIQ

      ; reset RTC-a
      STR R11,[R10,#0C]                 ; obrisi brojilo
      STR R11,[R10,#08]                 ; obrisi status (IACK)

      LDRB R0,[R9,#4]                   ; čitaj sa vrata B (OK je i LDR)

      BL DIJELI                         ; poziv

      STRB R0,[R9]                      ; pohrani podatak na vrata A (OK je i STR)

      LDMFD R13!,{R0,R1,R14}            ; kontekst
      SUBS PC,R14,#4                    ; povratak
; ------------------------------------------------------------------
DIJELI
      MOV R0,R0,LSR#2                   ; potprogram za dijeljenje s 4
      MOV PC,LR

; ------------------------------------------------------------------
GLAVNI
      MOV R13,#1<16           ; init stoga

      MOV R2, #1<8
      LDR R0,[R2],#4          ; adresa GPIO-a
      LDR R1,[R2],#4          ; adresa RTC-a

      MOV R2,#0FF
      STR R2,[R0,#8]          ; SMJER VRATA A
      STR R2,[R0,#0C]         ; SMJER VRATA B

      LDR R3,[R2]             ; brojac=20000(10)
      STR R3,[R1,#4]
      MOV R2,#0
      STR R2,[R1,#0C]         ; obrisi brojilo (nije nužno)

      MOV R2,#1
      STR R2,[R1,#10]         ; omoguci RTC prekid


      ; ukljuci FIQ prekid
      MRS R0,CPSR
      BIC R0,R0,#40
      MSR CPSR_c,R0           ; NORMAL MODE, FIQ enabled

LOOP  B LOOP                  ; petlja
      SWI 123456              ; nije potrebno


      'ORG 100                ; adrese

GPIO_ADDR DW 0FFFFFF00
RTC_ADDR  DW 0FFFFFE00
BROJAC    DW %D 20000
```

Završni ispit iz Arhitekture računala 1 29. lipnja 2009.

Prezime i ime (štampanim slovima):_____PREZIME IME_____ 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. Također izjavljujem da mi zdravstveno stanje dozvoljava pisanje ove zadaće. Potpis:_____.

Dozvoljeno je koristiti isključivo službene šalabahtere (popisi naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Ispit se piše 120 minuta.

Teoretski zadatci (ispunjavate na ovom papiru):

1. (*0,5 bodova*) Za procesor ARM napišite programski odsječak koji izvodi predznačno proširenje broja u registru R0 iz 24-bitnog zapisa 2'k, na 32 bita. Upotrijebite dvije naredbe MOV.

_____

_____

2. (*0,75 bodova*) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: _____, _____ i _____.

3. (*0,75 bodova*) Procesor ARM9 uvodi _____ arhitekturu memorijskog pristupa. Time se izbjegava _____ hazard. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave _____ hazarda.

4. (*1 bod*) Dvije osnovne metode predviđanja grananja su: _____ i _____ . Metoda kod koje se ispituje da li je adresa grananja _____ od PC-a te se tada pretpostavlja da će doći do grananja je _____ metoda predviđanja.

5. (*0,25 bodova*) Je li broj 204_{16} moguće upisati kao neposrednu vrijednost kod aritmetičke naredbe procesora ARM? Da ☐ - Ne ☐

6. (*0,25 bodova*) Na stog podatke spremamo podatke naredbom STMFD. Ako ih želimo pročitati sa stoga u iste registre trebamo koristiti naredbu: _____.

7. (*0,5 bodova*) Nastavak S u naredbi SUBS PC, R14, #4 znači da treba registar _____upisati u registar _____.

8. (*0,5 bodova*) Kod procesora FRISC, uvjetno izvođenje naredbi moguće je za _____ naredbe. Kod procesora ARM, uvjetno izvođenje naredbi moguće je za _____ naredbe.

9. (*3 boda*) Za sklop FRISC-PIO vrijedi (zaokružite točne tvrdnje – **Da** ili **Ne**, svaki točan odgovor je 0,25, a pogrešan -0,25 bodova):

u ulaznom načinu rada čita se cijeli bajt Da Ne	u načinu ispitivanja bitova neki se bitovi mogu čitati, a neki pisati Da Ne	u izlaznom načinu rada može postati spreman Da Ne	može postaviti zahtjev za prekid u izlaznom načinu rada Da Ne	može postati spreman u načinu postavljanja bitova Da Ne	može se slati maska tijekom inicijalizacije u načinu ispitivanja bitova Da Ne
upravljački registri ICR i OCR zauzimaju istu adresu Da Ne	nakon što u izlaznom načinu FRISC pošalje podatak PIO-u, PIO aktivira STROBE Da Ne	priključci READY i STROBE koriste se u ulaznom načinu Da Ne	prilikom inicijalizacije u načinu postavljanja bitova može se slati maska Da Ne	priključci READY i STROBE ne koriste se samo u načinu postavljanja bitova Da Ne	nakon što u izlaznom načinu rada PIO pošalje podatak vanjskom svijetu, postavlja stanje spremnosti Da Ne

10. (*1 bod*) Procesor FRISC u fazi dohvata dohvaća _____ (što) iz _____ (odakle) sa adrese koja je u adresni registar AR kopirana iz registra _____. Ono što je dohvatio, procesor sprema u registar _____.

11. (0,75 *bodova*) Kod FRISC-a se potprogram poziva naredbom _____, a kod ARM-a naredbom _____. Kod FRISC-a se povratna adresa iz potprograma sprema _____ (gdje), a kod ARM-a

se sprema _____ (gdje). Kod FRISC-a se povratak iz potprograma ostvaruje naredbom _____, a kod ARM-a naredbom _____.

12. (*1,5 bodova*) Bistabil stanja postoji u (zaokružite točne tvrdnje – **Da** ili **Ne**, svaki točan odgovor je 0,25, a pogrešan -0,25 bodova):

bezuvjetnim vanjskim jedinicama	uvjetnim vanjskim jedinicama	prekidnim vanjskim jedinicama	sklopu FRISC-CT	sklopu FRISC-PIO	sklopu FRISC-DMA
Da Ne	Da Ne	Da Ne	Da Ne	Da Ne	Da Ne

13. (*0,25 bodova*) Napišite primjer naredbe FRISC-a (treba napisati cijelu naredbu sa svim parametrima) koja koristi apsolutno procesorsko adresiranje i **zaokružite** u naredbi dio koji se odnosi na to adresiranje: _____.

14. (*0,5 bodova*) Zadan je podatak 6. Njegov prikaz u 5-bitnom formatu NBC izgleda ovako: _____ (prikažite binarno), a prikaz u 6-bitnom formatu 2'k izgleda ovako: _____ (prikažite binarno).

15. (*0,5 bodova*) Zadan je podatak -7. Njegov prikaz u 5-bitnom formatu 2'k izgleda ovako: _____ (prikažite binarno), a prikaz u 6-bitnom formatu s bitom za predznak izgleda ovako: _____ (prikažite binarno).

Rješenja teoretskih zadataka

1. (*0,5 bodova*) Za procesor ARM napišite programski odsječak koji izvodi predznačno proširenje broja u registru R0 iz 24-bitnog zapisa 2'k, na 32 bita. Upotrijebite dvije naredbe MOV.

MOV R0, R0 LSL #8

MOV R0, R0 ASR #8

2. (*0,75 bodova*) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: _____**OBRADU PODATAKA, AL NAREDBE**_____, _____**PRIJENOS PODATAKA, LOAD-STORE, MEMORIJSKE**_____ i _____**GRANANJE, UPRAVLJAČKE**_____.

3. (*0,75 bodova*) Procesor ARM9 uvodi __**HARVARDSKU**_____ arhitekturu memorijskog pristupa. Time se izbjegava _____**STRUKTURNI**_____ hazard. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave _____**PODATKOVNOG**_____ hazarda.

4. (*1 bod*) Dvije osnovne metode predviđanja grananja su: _____**STATIČKA**_____ i _____**DINAMIČKA**_____ . Metoda kod koje se ispituje da li je adresa grananja _____**MANJA**_____ od PC-a te se tada pretpostavlja da će doći do grananja je _____**STATIČKA** _____ metoda predviđanja.

5. (*0,25 bodova*) Je li broj 204_{16} moguće upisati kao neposrednu vrijednost kod aritmetičke naredbe procesora ARM? Da ☒ - Ne ☐

6. (*0,25 bodova*) Na stog podatke spremamo podatke naredbom STMFD. Ako ih želimo pročitati sa stoga u iste registre trebamo koristiti naredbu: _____**LDMFD, LDMIA**_____.

7. (*0,5 bodova*) Nastavak S u naredbi SUBS PC, R14, #4 znači da treba registar ___**SPSR**_____upisati u registar _____**CPSR**_____.

8. (*0,5 bodova*) Kod procesora FRISC, uvjetno izvođenje naredbi moguće je za _____**UPRAVLJAČKE**_____ naredbe. Kod procesora ARM, uvjetno izvođenje naredbi moguće je za _____**SVE ili SKORO SVE**_____ naredbe.

9. (*3 boda*) Za sklop FRISC-PIO vrijedi (zaokružite točne tvrdnje – **Da** ili **Ne**, svaki točan odgovor je 0,25, a pogrešan -0,25 bodova):

u ulaznom načinu rada čita se cijeli bajt	u načinu ispitivanja bitova neki se bitovi mogu čitati, a neki pisati	u izlaznom načinu rada može postati spreman	može postaviti zahtjev za prekid u izlaznom načinu rada	može postati spreman u načinu postavljanja bitova	može se slati maska tijekom inicijalizacije u načinu ispitivanja bitova
Da Ne	Da Ne	Da Ne	Da Ne	Da Ne	Da Ne
upravljački registri ICR i OCR zauzimaju istu adresu	nakon što u izlaznom načinu FRISC pošalje podatak PIO-u, PIO aktivira STROBE	priključci READY i STROBE koriste se u ulaznom načinu	prilikom inicijalizacije u načinu postavljanja bitova može se slati maska	priključci READY i STROBE ne koriste se samo u načinu postavljanja bitova	nakon što u izlaznom načinu rada PIO pošalje podatak vanjskom svijetu, postavlja stanje spremnosti
Da Ne	Da Ne	Da Ne	Da Ne	Da Ne	Da Ne

10. (*1 bod*) Procesor FRISC u fazi dohvata dohvaća _____**STROJNI KOD ili NAREDBU**_____ (što) iz _____**MEMORIJE**_____ (odakle) sa adrese koja je u adresni registar AR kopirana iz registra ____**PC**____. Ono što je dohvatio, procesor sprema u registar ___**IR**_____.

11. (0,75 *bodova*) Kod FRISC-a se potprogram poziva naredbom _____**CALL**_____, a kod ARM-a naredbom _____**BL**_____. Kod FRISC-a se povratna adresa iz potprograma sprema __**NA STOG**_____ (gdje), a kod ARM-a se sprema __**U LR ili R14**_____ (gdje). Kod FRISC-a se povratak iz potprograma ostvaruje naredbom _____**RET**_____, a kod ARM-a naredbom ___**MOV PC, LR ili MOV PC, R14**___.

12. (*1,5 bodova*) Bistabil stanja postoji u (zaokružite točne tvrdnje – **Da** ili **Ne**, svaki točan odgovor je 0,25, a pogrešan -0,25 bodova):

bezuvjetnim vanjskim jedinicama	uvjetnim vanjskim jedinicama	prekidnim vanjskim jedinicama	sklopu FRISC-CT	sklopu FRISC-PIO	sklopu FRISC-DMA
Da **Ne**	**Da** Ne	**Da** Ne	**Da** Ne	**Da** Ne	**Da** Ne

13. (*0,25 bodova*) Napišite primjer naredbe FRISC-a (treba napisati cijelu naredbu sa svim parametrima) koja koristi apsolutno procesorsko adresiranje i **zaokružite** u naredbi dio koji se odnosi na to adresiranje: **npr. LOAD R0, (1000) ili STORE, JP, CALL**.

14. (*0,5 bodova*) Zadan je podatak 6. Njegov prikaz u 5-bitnom formatu NBC izgleda ovako: _____**00110**___ (prikažite binarno), a prikaz u 6-bitnom formatu 2'k izgleda ovako: ____**000110**_____ (prikažite binarno).

15. (*0,5 bodova*) Zadan je podatak -7. Njegov prikaz u 5-bitnom formatu 2'k izgleda ovako: __**11001**_____ (prikažite binarno), a prikaz u 6-bitnom formatu s bitom za predznak izgleda ovako: ___**100111**_____ (prikažite binarno).

Programski zadatci (rješavate na svojim papirima):

16. (*5 bodova*) U računalnom sustavu nalaze se FRISC, DMA jedinica (na adresi $FFFF1000_{16}$), bezuvjetna vanjska jedinica (na adresi $FFFF2000_{16}$) i uvjetna vanjska jedinica (na adresi $FFFF3000_{16}$). Potrebno je krađom ciklusa prenijeti 1000_{10} podataka s bezuvjetne vanjske jedinice u memorijski blok koji počinje na adresi 500_{16}. Dojavu kraja prijenosa potrebno je riješiti preko prekidnog potprograma (DMA je spojen na INT0).

Glavni program cijelo vrijeme treba izvoditi praznu petlju. U prekidnom potprogramu treba nakon završetka prijenosa na uvjetnu vanjsku jedinicu poslati broj $123456AB_{16}$. Čuvajte stanja registara.

```
DMASRC   `EQU 0FFFF1000
DMADEST  `EQU 0FFFF1004
DMACNT   `EQU 0FFFF1008
DMACTRL  `EQU 0FFFF100C
DMAST    `EQU 0FFFF1010
DMAIACK  `EQU 0FFFF1014
BVJ      `EQU 0FFFF2000
BVJSTOP  `EQU 0FFFF2004
UVJ      `EQU 0FFFF3000
UVJST    `EQU 0FFFF3004              ; konstante

         `ORG 0
         MOVE    10000, SP          ; stog
         JP      GLAVNI             ; skok na glavni zbog p. vektora

         `ORG 8                     ; vektor
         DW      IRQ                ; adresa pp, ne skok

GLAVNI
         MOVE    BVJ, R0            ; adresa bezuvjetne
         STORE   R0, (DMASRC)       ; pocetna adresa DMA

         MOVE    500, R0            ; adresa bloka
         STORE   R0, (DMADEST)      ; odrediste DMA
```

```
        MOVE    %D 1000, R0         ; broj podataka
        STORE   R0, (DMACNT)        ; u DMA

        MOVE    %B 0111, R0         ; iz VJ u MEM, kradja ciklusa, prekidi
        STORE   R0, (DMACTRL)       ; podesi

        MOVE    %B 10010000, SR     ; GIE, EINT0
        STORE   R0, (DMAST)         ; pokreni DMA

PET     JP      PET                 ; prazna petlja

        HALT                        ; ne treba


        `ORG    ...                 ; ako je tu broj, mora biti isti kao na ORG 8
IRQ     PUSH    R0                  ; kontekst ukupno
        MOVE    SR, R0
        PUSH    R0
        PUSH    R1

        STORE   R0, (DMAIACK)       ; dojavi prekid

        LOAD    R1, (PODATAK)       ; ucitaj podatak

CEK     LOAD    R0, (UVJST)         ; cekaj spremnost UVJ
        CMP     R0, 0
        JP_Z    CEK                 ; ili JP_EQ

        STORE   R1, (UVJ)           ; posalji podatak
        STORE   R1, (UVJST)         ; pobriši status

        POP     R1                  ; kontekst
        POP     R0
        MOVE    R0, SR
        POP     R0
        RETI                        ; povratak iz pp

PODATAK
        DW      123456AB            ; konstanta
                                    ; (prevelika za MOVE)

        `ORG 500                    ; blok (ne treba)
BLOK    DW ...
```

17. (*4 boda*) Za procesor ARM napisati program koji inicijalizira RTC sklop tako da generira IRQ prekide svakih 3,5 sekundi. Na RTC je spojen signal frekvencije 10kHz.

U prekidnom potprogramu za IRQ pročitajte stanje na vratima B GPIO sklopa i ako je bilo koja linija vrata postavljena (različita od 0) tada upišite vrijednost 1 na lokaciju 1000_{16}, a ako su sve linije vrata B jednake 0 tada upišite vrijednost 0 na lokaciju 1000_{16}. Sve potrebne inicijalizacije RTC i GPIO sklopova obavite u glavnom programu. Adrese sklopova odaberite po želji.

Nakon inicijalizacije glavni program izvodi praznu petlju. Čuvajte stanja registara.

```
        `ORG 0
        B     GLAVNI                ; skok na glavni zbog vektora

        `ORG 18                     ; vektor
        B     IRQ                   ; skok ili direktno potprogram

IRQ     STMFD SP!, {R0,R1,R2,R3}        ; kontekst

        ; ucitaj si adrese ponovno
        MOV   R0, #2<12             ;
        LDR   R1, [R0], #4          ;
        LDR   R2, [R0], #4          ;
```

```
        LDR    R0, [R2, #4]        ; procitaj GPIO B
        CMP    R0, #0              ; provjeri jesu li sve linije 0
        ; u slucaju da su sve linije 0, ne treba ponovno stavljati 0 u registar


        BEQ    DALJE               ; uvjetno izvođenje    ----    verzija 1
        MOV    R0, #1              ; podatak 1
DALJE

        ;ili
        MOVNE R0, #1               ; podatak 1          ---- verzja 2 (kraće)


        MOV    R3, #1<12           ; adresa
        STR    R0,   [R3]          ; spremi na adresu

        STR    R0, [R1, #8]        ; obrisi RTCINT

        MOV    R0, 0               ; brojac na 0
        STR    R0, [R1, #0C]       ; RTCLR

        MOV    R0, #2<12           ; ucitaj adrese
        LDR    R3, [R0, #0C]       ; ucitaj konstantu sa KONST
        STR    R3, [R1, #4]        ; napuni ponovno RTCMR (ne treba)

        LDMFD SP!, {R0,R1,R2,R3}        ; kontekst
        SUBS   PC, LR, #4          ; povratak iz pp

GLAVNI
        MOV    SP,    #10<12       ; stog
        MOV    R0, #2<12           ; adrese u mem

        LDR    R1, [R0], #4        ; R1 = RTC
        LDR    R2, [R0], #4        ; R2 = GPIO

        LDR    R3, [R0]            ; R3 = konst = 35000
        STR    R3, [R1, #4]        ; konstanta u RTCMR

        MOV    R3, #0
        STR    R3, [R1, #0C]       ; obriši brojilo (ne treba)

        MOV    R3, #1
        STR    R3, [R1, #10]       ; RTCCR = 1, prekidi

        MOV    R3, #0FF            ; sve linije ulazne
        STR    R3, [R2, #0C]       ; smjer GPIO B

        MRS    R0, CPSR            ; omoguci prekide
        BIC    R0, R0, #80         ; 80!
        MSR    CPSR_c, R0

PET     B      PET                 ; prazna petlja

        SWI    123456              ; ..ili HALT (ne treba)

        `ORG   2000
RTC     DW     2100                ; adrese
GPIO    DW     2200
KONST DW %D 35000                  ; konstanta
```

18. (*3 boda*) Napišite program za procesor ARM koji inicijalizira brze prekide (FIQ) i nakon toga izvodi praznu petlju. Napišite prekidni potprogram za FIQ. Prekidni potprogram treba na adresu 1000_{16} zapisati 8-bitni podatak 1. Čuvajte stanja registara.

```
    `ORG 0
    B MAIN                      ; skok na glavni
```

```
    `ORG 1C                 ; adresa potprograma za FIQ

    STMFD SP!, {R0, R1}     ; kontekst

    MOV R0, #1<12           ; adresa 1000
    MOV R1, #1              ; podatak 1
    STRB R1, [R0]           ; spremi 8 bita

    LDMFD SP!, {R0, R1}     ; vraćanje konteksta
    SUBS PC, LR, #4         ; povratak iz potprograma


MAIN
    MOV R13, #1<12          ; stog

    MRS R0, CPSR            ; statusni registar
    BIC R0, R0, #40         ; ovdje je 40, ne 80!!!
    MSR CPSR_c, R0          ; vrati status
PET
    B   PET                 ; prazna petlja
```

19. (*6 bodova*) Za procesor ARM treba napisati potprogram TEZ_SUM koji računa težinsku sumu niza. Potprogram prima tri parametra: adresu niza podataka (R0), adresu niza težina (R1), te duljinu niza koja je zajednička za oba niza (R2). Potprogram treba pomnožiti vrijednosti na istim pozicijama u oba niza, te zbrojiti umnoške u jedan rezultat. Rezultat potprograma vraća se u registru R0. Svi podaci su u 8-bitnom 2'k formatu, dok rezultat treba biti u 32-bitno 2'k formatu. Pretpostavka je da kod množenja i zbrajanja neće doći do prekoračenja opsega od 32 bita. Potprogram treba čuvati stanja registara koje koristi.

Primjer:	niz podataka je	1, 2, 3, -5, 8
	niz težina je	6, 3, -1, 4, 7
	duljina nizova je	5

Rezultat potprograma je: $1*6 + 2*3 + 3*(-1) + (-5)*4 + 8*7 = 45_{10}$

```
; R0 - adresa podataka
; R1 - adresa tezina
; R2 - brojac

TEZ_SUM
      STMFD SP!, {R1, R2, R3, R4, R5, R6} ; kontekst – R1 i R2 su parametri, ni njih se ne smije mijenjati

      MOV   R5, #0                        ; inicijalizacija
      MOV   R6, #0

PET   LDRSB R3, [R0], #1                  ; ucitaj i p. prosiri
      LDRSB R4, [R1], #1                  ;

      ; pomnozi zbroji i skuhaj kavu      ;
      SMLAL R5, R6, R3, R4                ; verzija 1
      ;ili
      MLA   R5, R3, R4, R5                ; verzija 2
      ;ili
      MUL   R6, R3, R4                    ; verzija 3
      ADD   R5, R6, R5                    ;
      ; ili ako se bas mora, mnozenje uzastopnim pribrajanjem...  verzija 4

      SUBS  R2, R2, #1                    ; petlja
      BNE   PET                           ; skok

KRAJ  MOV   R0, R5                        ; pripremi rezultat

      LDMFD SP!, {R1, R2, R3, R4, R5, R6} ; kontekst
      MOV   PC, LR                        ; povratak
```

Završni ispit iz ARH 1 — 28. lipnja 2010.

Prezime i ime (velikim 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 FRISC-a). Programe treba pisati uredno i komentirati pojedine cjeline programa.

1.a. (*0,4 boda*) Broj -5 treba zapisati u 16-bitnom formatu 2'k u memoriju FRISC-a od adrese 100_{16}. U bajtu na adresi 100_{16} pisat će ____FB____, a u bajt na adresi 101_{16} pisat će ____FF_____ (prikažite bajtove u heksadekadskoj bazi).

1.b. (*1 bod*) S obzirom na smještaj operanada postoje procesorske arhitekture: _____stogovna _____, _____akumulatorska_____, ____registar-memorija ___ i _____registar-registar (ili load-store)_____.
Za RISC procesore, od ove 4 arhitekture uobičajeno se koristi _____registar-registar _____.

1.c. (*1 bod*) Sabirnice se prema načinu komunikacije dijele na _____sinkronu __ i __asinkronu _____. Po toj podjeli, sabirnica FRISC-a je _____sinkrona _____. Prilagodba brzine komunikacije ostvaruje se pomoću FRISC-ovog priključka _____WAIT _____ koji je po smjeru _____ulazni _____.

1.d. (*1 bod*) Tri općenite faze izvođenja naredbe su **redom** ___dohvat _____, ___dekodiranje __ i __izvođenje _____. Protočna struktura FRISC-a ima __2__ (koliko) razine. Druga po redu općenita faza izvođenja je kod FRISC-a smještena u _prvu_ (koju po redu) razinu.

1.e. (*1 bod: ispravan 0,2, pogrešan -0,2, prazno 0*) Za sklop FRISC-CT uz svaku tvrdnju zaokružite Točno ili Netočno

Brojilo DC u CT-u broji prema gore - od nule do vrijednosti u registru LR	Brojilo u CT-u se smanjuje kad se pojavi impuls na ulaznom priključku CNT	Kad CT postane spreman, onda se uvijek generira prekid	Kad CT postane spreman, onda se uvijek generira impuls na priključku ZC	Kad CT postane spreman, onda se u brojilo DC automatski napuni vrijednost iz registra LR
Točno **Netočno**	**Točno** Netočno	Točno **Netočno**	**Točno** Netočno	**Točno** Netočno

2.a. *(1 bod)* Nakon uključenja procesor ARM 7 treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:_12

```
    `ORG 0
    MOV R0, #5          3
    EOR R0,R0,R0        1
    ADD R0,R0,#2        1
A   SUBS  R0,R0,#1      1  1
    BNE A               3  1
    EOR R0,R0,R0           1
```

2.b. *(0,5 bodova)* Neposredna vrijednost kod aritmetičko-logičkih naredaba procesora ARM može se zapisati kao broj širine ___8_____ bitova koji se rotira u desno za _PARAN___ broj bitova.
2.c. *(0,25 bodova)* Koji sklop kod procesora ARM omogućuje da se drugi operand pomakne ili rotira za proizvoljan broj bitova prije aritmetičko-logičkih operacije: _____BARREL-SHIFTER_____.
2.d. *(0,5 bodova)* Nabrojite dvije naredbe za pristup registrima stanja procesora ARM: _MRS_, _MSR_.

2.e. *(0,75 bodova)* Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak __88776655__(0,5)_____, a u registru R1 podatak _100_(0,25)__ .

```
`ORG 0
MOV   R1,#1<8
LDR   R0,[R1,#4]
`ORG  100
DB 11, 22, 33, 44, 55, 66, 77, 88, 99, AA
```

2.f. *(1 bod)* Neka priručna memorija ima 32_{10} bloka i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
0x2	2
0x21	1
0x22	2
0x153	0x13 ILI 19_{10}

2.g. *(0,75 bodova)* Kod ARM-a, GPIO i RTC se spajaju na sabirnicu __APB_. Memorija i procesor ARM se spajaju sa sabirnicom _____AHB_____. Između ovih dvaju sabirnica nalazi se sklop koji se zove__MOST (APB-AHB MOST)_____ .

2.h. *(1 bod)* Dvije osnovne metode predviđanja grananja su: _____STATIČKA_____ i _____DINAMIČKA_____. Metoda kod koje se ispituje da li je adresa grananja _____MANJA____ od PC-a te se tada pretpostavlja da će doći do grananja je _____STATIČKA_____ metoda predviđanja.

2.i. *(0,25 bodova)* Podatkovni hazard može se javiti na arhitekturi ARM 9. Dopunite naredbu tako da dođe do podatkovnog hazarda: ADD R1, R2, R3
SUB R5, _R1_, R7

2.j. *(0,2 boda)* Na stog podatke spremamo podatke naredbom STMFD. Ako ih želimo pročitati sa stoga u iste registre trebamo koristiti naredbu: ____LDMFD (ILI LDMIA)_____.

3. (*4,2 boda*) Za procesor FRISC napisati **potprogram CTO** koji broji početne jedinice u 16-bitnom podatku. Adresa 16-bitnog podatka se u potprogram prenosi stogom, a uklanja je pozivatelj. Broj početnih jedinica se vraća preko R0. U potprogramu treba čuvati stanja registara.

U glavnom programu treba pozvati potprogram CTO za svaki podatak iz memorijskog bloka 16-bitnih podataka. Blok počinje na adresi 1000_{16} i sadrži 100_{10} slijednih 16-bitnih podataka (bez praznina između). Na kraju programa treba na lokaciju SUMA zapisati ukupni zbroj početnih jedinica svih podataka iz bloka.

4. (*2,0 boda*) Za procesor ARM napisati **odsječak** programa koji **sa stoga učitava 64-bitni broj** zapisan u formatu 2'k, zatim **uspoređuje dva 32-bitna broja** koji se nalaze **u R0 i R1** te **ako su isti** njihov **umnožak pribraja 64-bitnom broju** i **pohranjuje ga** natrag **na isto mjesto na stog.** Pretpostavka: stog je inicijaliziran i podaci su već na stogu i u registrima.

5. (*3,0 boda*) Za procesor ARM uporabom sklopa GPIO napisati **3 potprograma** za upravljanje svjetlima vozačkog i pješačkog semafora na pješačkom prijelazu. Na vrata B sklopa GPIO spojena je redom signalizacija svjetla semafora kao u tablici. Slanje na pin „0“označava gašenje svjetla, a „1“ paljenje. Potprogrami trebaju čuvati stanja registara.

Vrata B sklopa GPIO	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1	Pin 0
Spojena signalizacija	Zeleno za pješake	Crveno za pješake	Nije spojeno	Nije spojeno	Nije spojeno	Zeleno za vozače	Žuto za vozače	Crveno za vozače

Postoje 3 potprograma koja treba napisati:

- Potprogram ZELENO – zeleno svjetlo vozačima, crveno svjetlo pješacima
- Potprogram ZUTO – žuto svjetlo vozačima, crveno svjetlo pješacima
- Potprogram CRVENO – crveno svjetlo vozačima, zeleno svjetlo pješacima

Napisati i **dio glavnog programa** za inicijalizaciju sklopa GPIO. Adresu sklopa odaberite proizvoljno.

6. (*7,6 bodova*) Za procesor ARM napisati **program i prekidni potprogram** koji rješava problem semafora pješačkog prijelaza uporabom sklopa RTC. Za paljenje svjetala semafora trebate iskoristiti potprograme iz prethodnog zadatka. Adrese sklopova odaberite proizvoljno. Na RTC je spojen signal frekvencije 10Hz. Glavni program izvodi praznu petlju.

Sklop RTC generira prekid nakon proteka perioda vremena za određeno svjetlo. U prekidnom potprogramu treba provjeravati u kojem stanju se semafor nalazio, prijeći u sljedeće stanje i postaviti novu kombinaciju svjetala te period čekanja. Trenutno stanje treba svaki put nakon promjene pohraniti na memorijsku lokaciju STANJE. Postoje 4 stanja koja se mijenjaju slijedno, a nakon što semafor dođe u stanje 3, sljedeće stanje je ponovno 0.

STANJE	**SLJEDEĆE STANJE**	**AKCIJA**	**KORIŠTENI POTPROGRAM ZA SVJETLA**
0	1	3 minute gori zeleno svjetlo za vozače, a crveno za pješake	potprogram ZELENO
1	2	5 sekundi gori žuto svjetlo za vozače, a crveno za pješake	potprogram ZUTO
2	3	30 sekundi gori crveno svjetlo za vozače, a zeleno za pješake	potprogram CRVENO
3	0	3 sekunde gori žuto svjetlo za vozače, a crveno za pješake	potprogram ZUTO

7. (*2,6 bodova*) Na linije brzog prekida (FIQ) procesora ARM spojen je gumb za pješake na semaforu iz prethodnog zadatka, kojim pješaci „traže brži prelazak ulice“. Kao dodatak prethodnom zadatku, potrebno je napisati **prekidni potprogram** kojim se trenutno prekida izvođenje ciklusa rada semafora i prelazi u STANJE = 1 (žuto svjetlo za vozače, a crveno za pješake) koje traje 5 sekundi. Ciklus se dalje normalno nastavlja kao u prethodnom zadatku (crveno vozači, zeleno pješaci). Pretpostavite da gumb radi samo kad je stanje semafora 0 (zeleno vozači, crveno pješaci).
Treba napisati i **dio glavnog programa** koji omogućava FIQ bez da naruši ponašanje programa iz prethodnog zadatka.

3) RJEŠENJE

```
CTO     PUSH R1                     ; spremanje konteksta
        LOAD R1, (SP+8)             ; dohvat parametra
        LOADH R1, (R1)              ; dohvat podatka
        ROTL R1, %D 16, R1          ; eliminacija gornjih 16 bita
        MOVE 0, R0                  ; inicijalizacija rezultata
LOOP    ROTL R1, 1, R1              ; rotacija u petlji
        JR_NC VAN                   ; ispitivanje bita
        ADD R0, 1, R0               ; povećavanje rezultata
        JR LOOP                     ; petlja
VAN     POP R1
        RET                         ; povratak

GLAVNI MOVE   10000, SP             ; init stoga
       MOVE   1000, R1              ; početak bloka
       MOVE   0, R2                 ; brojač početnih jedinica
PETLJA PUSH   R1                    ; adresa podatka na stog
       CALL   CTO                   ; pozovi potprogram
       ADD    SP, 4, SP             ; ukloni sa stoga adresu
       ADD    R2, R0, R2            ; pribroji jedinice
       ADD    R1, 2, R1             ; pomak na sljedeći
       CMP    R1, 10C8              ; provjeri kraj
       JP_NE  PETLJA                ; ako ne skoči na oduzmi
       STORE  R2, (SUMA)            ; spremanje brojača
KRAJ   HALT
```

4) RJEŠENJE

```
        LDMFD         SP!, {R2, R3}
        CMP           R0, R1
        SMLALEQ       R2, R3, R0, R1
        STMFDEQ       SP!, {R2, R3}
```

5) RJEŠENJE

```
ZELENO STMFD R13!, {R0, R1, R14}  ; pohrani kontekst
       LDR    R0, GPIO            ; R0 = GPIO
       MOV    R1, #%B 01000100    ; uključi zeleno vozači, crveno pješaci
       STR    R1, [R0, #4]        ; pošalji na GPIO B
       LDMFD  R13!, {R0, R1, R14} ; obnovi kontekst
       MOV    PC, LR              ; povratak

ZUTO   STMFD R13!, {R0, R1, R14}  ; pohrani kontekst
       LDR    R0, GPIO            ; R0 = GPIO
       MOV    R1, #%B 01000010    ; uključi žuto vozači, crveno pješaci
       STR    R1, [R0, #4]        ; pošalji na GPIO B
       LDMFD  R13!, {R0, R1, R14} ; obnovi kontekst
       MOV    PC, LR              ; povratak

CRVENO STMFD R13!, {R0, R1, R14}  ; pohrani kontekst
       LDR    R0, GPIO            ; R0 = GPIO
       MOV    R1, #%B 10000001    ; uključi crveno vozači, zeleno pješaci
       STR    R1, [R0, #4]        ; pošalji na GPIO B
       LDMFD  R13!, {R0, R1, R14} ; obnovi kontekst
       MOV    PC, LR              ; povratak

GLAVNI ...    (može i bez toga, ali to treba navesti u programu)
       LDR    R0, GPIO            ; R0 = GPIO
       MOV    R1, #%B 00000000    ; izlazne linije
       STR    R1, [R0, #0C]       ; smjer GPIO B
       ...
GPIO   DW     0FFFFFE00           ; adresa GPIO
```

6) RJEŠENJE

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

        `ORG    18                      ; adresa prek. potprograma
        STMFD R13!, {R0, R1, R2, R14}     ; pohrani kontekst
        LDR     R0, RTC                 ; adresa RTC
        LDR     R1, STANJE              ; učitaj STANJE

        CMP     R1, #0                  ; ako je stanje bilo 0 (zeleno)
        BLEQ    ZUTO                    ; pozovi potprogram ZUTO
        MOVEQ   R2, #%D 50              ; učitaj 50
        B       VAN

        CMP     R1, #1                  ; ako je stanje bilo 1 (prvo žuto)
        BLEQ    CRVENO                  ; pozovi potprogram CRVENO
        MOVEQ   R2, #%D 300             ; učitaj 300
        B       VAN

        CMP     R1, #2                  ; ako je stanje bilo 2 (crveno)
        BLEQ    ZUTO                    ; pozovi potprogram ZUTO
        MOVEQ   R2, #%D 30              ; učitaj 30
        B       VAN

        CMP     R1, #3                  ; ako je stanje bilo 3 (drugo žuto)
        BLEQ    ZELENO                  ; pozovi potprogram ZELENO
        MOVEQ   R2, #%D 1800            ; učitaj 1800

VAN     STR     R2, [R0, #4]            ; konstanta u RTCMR
        ADD     R1, R1, #1              ; povećaj STANJE za 1 (ciklus)
        CMP     R1, #4                  ; ako je 4 (nepostojeće) onda
        MOVEQ   R1, #0                  ; vrati na STANJE = 0
        STR     R1, STANJE              ; pohrani STANJE
        STR     R1, [R0, #8]            ; obrisi RTCINTR
        MOV     R1, #0                  ; obriši RTCCLR
        STR     R1, [R0, #0C]
        LDMFD   R13!, {R0, R1, R2, R14}    ; obnovi kontekst
        SUBS    PC, LR, #4              ; povratak

GLAVNI MOV      SP, #1<16               ; stog
        LDR     R0, RTC                 ; adresa RTC
        MOV     R1, #%D 1800            ; inicijalno čekanje KONST1 = 1800
        STR     R1, [R0, #4]            ; konstanta u RTCMR
        MOV     R1, #0                  ; obriši RTCCLR (nije nužno)
        STR     R1, [R0, #0C]
        MOV     R1, #1
        STR     R1, [R0, #10]           ; RTCCR = 1, prekid
        MRS     R0, CPSR                ; omogući prekide
        BIC     R0, R0, #80             ; 80
        MSR     CPSR_c, R0
        LDR     R0, GPIO                ;  R0 = GPIO (iz prethodnog zadatka, ne boduje se)
        MOV     R1, #%B 00111000        ; izlazne linije (iz prethodnog zadatka, ne boduje se)
        STR     R1, [R0, #0C]           ; smjer GPIO B (iz prethodnog zadatka, ne boduje se)
        MOV     R1, #0                  ; inicijalno STANJE = 0 (zeleno) - nije nužno ako DW 0
        STR     R1, STANJE              ; pohrani STANJE
        BL      ZELENO                  ; uključi zeleno
PET     B       PET                     ; prazna petlja

RTC     DW      0FFFFFF00               ; adresa RTC
GPIO    DW      0FFFFFE00               ; adresa GPIO
STANJE DW       0                       ; stanje semafora
```

7) RJEŠENJE

```
        `ORG   1C                   ; adresa prek. potprograma FIQ
        STMFD R13!, {R0, R1}        ; pohrani kontekst
        LDR    R0, RTC              ; adresa RTC
        STR    R1, [R0, #8]         ; obrisi RTCINTR
        BL     ZUTO                 ; pozovi potprogram ZUTO
        MOV    R1, #%D 50           ; 50
        STR    R1, [R0, #4]         ; konstanta u RTCMR
        MOV    R1, #1               ; STANJE = 1
        STR    R1, STANJE           ; pohrani STANJE
        MOV    R1, #0               ; obriši RTCCLR
        STR    R1, [R0, #0C]
        LDMFD  R13!, {R0, R1}       ; obnovi kontekst
        SUBS   PC, LR, #4           ; povratak

GLAVNI ...
        MRS R0, CPSR              ; statusni registar
        BIC R0, R0, #C0           ; omogući IRQ i FIQ (40 i 80)
        MSR CPSR_c, R0            ; vrati status
```

Završni ispit iz ARH 1 **17. lipnja 2011.**

Prezime i ime (velikim 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 popise naredaba FRISC-a i ARM-a. Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir.

1. (2 boda) **1.a.** 1011_2 je 4-bitni prikaz nekog broja X u formatu dvojnog komplementa. Broj X iznosi: _____. Prikažite broj X u 6-bitnom formatu dvojnog komplementa: _____. Prikažite broj X u 8-bitnom formatu s bitom za predznak _____

1.b. 0011_2 je 4-bitni prikaz nekog broja Y u formatu dvojnog komplementa. Broj Y iznosi: _____.
Prikažite broj Y u 6-bitnom formatu dvojnog komplementa: _____. Prikažite broj Y u 8-bitnom formatu s bitom za predznak _____. Prikažite broj Y u 6-bitnom NBC formatu: _____.

1.c. 0011_2 je 4-bitni prikaz nekog broja Z u NBC formatu. Broj Z iznosi: _____. Prikažite broj Z u 6-bitnom prikazu dvojnog komplementa: _____. Prikažite broj Z u 8-bitnom formatu s bitom za predznak_____.

2. (1 bod) Harvardska arhitektura ima razdvojenu _____ i _____. Glavna prednost Harvardske (u odnosu na Von Neumannovu arhitekturu) je _____. Prednost Von Neumannove arhitekture je _____.

3. (3 boda) Četiri načina rada sklopa FRISC-PIO su: _____, _____, _____ i _____. PIO se na vanjski proces spaja pomoću priključaka _____ širine _____ bita te pomoću dva jednobitna priključka _____ i _____ koji služe za _____. Ova dva priključka koriste se u načinima rada _____ i _____. PIO **ne može** postati spreman kad radi u načinu rada _____.

4. (1,5 boda) Potprogram se kod FRISC-a poziva naredbom _____, a kod ARM-a naredbom _____. Povratna adresa se kod FRISC-a sprema _____(gdje), a kod ARM-a _____. Povratak iz potprograma se kod FRISC-a ostvaruje naredbom _____, a kod ARM-a naredbom _____.

5.a. (1,75 boda) Vrste prekida kod FRISC-a su _____(A) i _____(B). Adresa prekidnog potprograma (A) je _____, a za prekid (B) adresa potprograma je _____
Za oba prekida, povratna adresa se sprema _____(gdje). Povratak iz prekidnog potprograma (A) ostvaruje se naredbom _____, a iz (B) pomoću naredbe _____.

5.b. (2,25 boda) Kod ARM-a imamo prekide _____(C) i _____(D). Adresa prekidnog potprograma (C) je _____, a za prekid (D) adresa potprograma je _____. Za prekid (C) povratna adresa se sprema _____(gdje), a za prekid (D) povratna adresa se sprema _____(gdje). Povratak iz prekidnog potprograma (C) ostvaruje se naredbom _____, a iz (D) pomoću naredbe _____. ARM kod prihvaćanja prekida, osim povratne adrese, automatski sprema još i registar _____.

6. (2 boda) Na sabirnici AHB čitanje iz **brze** memorije podijeljeno je na _____ fazu koja traje _____ takta clock-a i na _____ fazu koja traje _____ takta clock-a. Zbog preklapanja ovih faza, efektivno će **četiri** uzastopna brza čitanja trajati _____ taktova clock-a. Podatkovni priključci procesora ARM ukupno su široki _____ bita, od čega jedna polovica ima _____ smjer, a druga polovica _____ smjer.

7. (1 bod) Neka priručna memorija ima 48_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	**Adresa bloka u priručnoj memoriji**
0x23	
0x48	
0x59	
0x90	

OKRENI STRANICU!

8. (*5,5 bodova*) Za procesor FRISC napisati **potprogram SALJI** koji preko sklopa DMA šalje podatke iz memorije na bezuvjetnu vanjsku jedinicu BVJ. Adresu početka bloka podataka i broj podataka potprogram prima preko stoga. DMA radi zaustavljanjem procesora. Potprogram treba čuvati kontekst, a parametre sa stoga treba obrisati pozivatelj.

U glavnom programu treba **učitavati 32-bitne podatke** s dvije nezavisne uvjetne vanjske jedinice, UVJ1 i UVJ2. Podatke treba **spremati** u memoriju, počevši od adrese PODACI1 (*stigli s UVJ1*) i PODACI2 (*stigli s UVJ2*). Kada je s obje vanjske jedinice zajedno **primljeno ukupno 99_{10} podataka**, treba potprogramom SALJI na BVJ **prenijeti blok (PODACI1 ili PODACI2)** u kojemu se nalazi **više podataka.** Nakon toga treba zaustaviti procesor. Adrese sklopova odaberite proizvoljno.

9. (4 *boda*) U memoriji procesora ARM na lokaciji 1000_{16} se nalazi **niz** od 20_{16} **parova 8-bitnih brojeva** u formatu **2'k**, gdje su brojevi u pojedinom paru zapisani neposredno jedan iza drugog. Napisati glavni program, koji pomoću potprograma PAROVI treba obraditi ovaj blok podataka.

Napisati potprogram PAROVI koji preko fiksne lokacije POCETAK prima adresu bloka memorije, a preko fiksne lokacije BROJ prima broj parova 8-bitnih brojeva u bloku, zapisanih u formatu 2'k. Potprogram treba za svaki par u bloku usporediti brojeve u pojedinom paru i veći od njih poslati potprogramu ABS kao 32-bitni broj. Rezultat potprograma ABS treba kao jedan 16-bitni broj pohraniti na ista dva bajta koja je zauzimao odgovarajući par brojeva.

Napisati potprogram ABS koji treba izračunati **apsolutnu vrijednost 32-bitnog broja** zapisanog u **formatu 2'k**. Potprogram prima parametar preko stoga, a rezultat vraća preko registra.
Potprogrami trebaju čuvati kontekst, a parametre sa stoga treba brisati pozivatelj.

10. (*6 bodova*) Procesor ARM pomoću sklopa GPIO upravlja pametnom mikrovalnom pećnicom. Ovisno o **vrsti i masi hrane**, pećnica sama određuje **trajanje pečenja**. Pinovi **ulaznih vrata A sklopa GPIO** označavaju sljedeće:

- **bitovi [3:0]** – masa hrane zaokružena na kvante od 100 g (*npr. broj 4 = 400 g*)
- **bitovi [7:4]** – vrsta hrane (od 0 do 15)

Pinovi **vrata B sklopa GPIO** označavaju sljedeće:

- **bit 0** – ulazni signal READY, kojim pećnica dojavljuje procesoru da su podaci o masi i vrsti hrane uneseni i da je pećnica spremna (slanjem vrijednosti 1)
- **bit 1** – izlazni signal STROBE, kojim procesor dojavljuje pećnici da je primio podatke (slanjem impulsa 1). Tada će pećnica automatski deaktivirati signal READY
- **bit 2** – izlazni signal za uključivanje pećnice (vrijednost 1) ili isključivanje pećnice (vrijednost 0)
- ostali bitovi se ne koriste, ali ih je potrebno postaviti **ulaznima**

Svaka od 16 vrsta hrane ima svoje **specifično trajanje pečenja za količinu od 100 grama**, a trajanje je **zadano u minutama**. Ukupno trajanje pečenja dobiva se **množenjem** mase hrane sa specifičnim trajanjem rada za tu vrstu hrane. Specifična trajanja rada dana su u tablici velikoj 16 bajtova i zapisanoj u memoriji na adresi **1000_{16}**. Svaki **bajt u tablici** označava **specifično trajanje rada za pojedinu vrstu** hrane, pri čemu je redoslijed hrane u tablici od vrste 0 na adresi **1000_{16}** do vrste 15 na adresi **$100F_{16}$**.

U glavnom programu, procesor čeka da pećnica dojavi pomoću READY da je spremna za pečenje. Tada procesor pročita podatke i impulsom na STROBE dojavljuje da je primio podatke i nakon toga uključuje pećnicu. Također pokreće **sklop RTC**, koji **mjeri vrijeme rada** mikrovalne pećnice. Po isteku potrebnog vremena rada za navedenu vrstu i masu hrane, RTC koji je spojen na **signal FIQ**, postavlja zahtjev za prekid. U prekidnom programu treba **isključiti pećnicu**, te se vratiti u glavni program i ponovno čekati da pećnica bude spremna. Adrese sklopova odaberite proizvoljno. Na RTC je spojen signal frekvencije 10Hz.

Mem. lokacija	Vrijednost	Opis
1000_{16}	1_{16}	100 g vrste hrane 0000 kuha se 1 min
1001_{16}	5_{16}	100 g vrste hrane 0001 kuha se 5 min
. . .	. . .	. . .
$100F_{16}$	3_{16}	100 g vrste hrane 000F kuha se 3 min

Završni ispit iz ARH 1 **17. lipnja 2011.**

Prezime i ime (velikim 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 popise naredaba FRISC-a i ARM-a. Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir.

1. (2 boda) **1.a.** 1011_2 je 4-bitni prikaz nekog broja X u formatu dvojnog komplementa. Broj X iznosi: __**-5**___. Prikažite broj X u 6-bitnom formatu dvojnog komplementa: ___**111011**__. Prikažite broj X u 8-bitnom formatu s bitom za predznak__**10000101**___.

1.b. 0011_2 je 4-bitni prikaz nekog broja Y u formatu dvojnog komplementa. Broj Y iznosi: ___**+3**____. Prikažite broj Y u 6-bitnom formatu dvojnog komplementa: _____**000011**_____. Prikažite broj Y u 8-bitnom formatu s bitom za predznak_____**00000011**_____. Prikažite broj Y u 6-bitnom NBC formatu: _____**000011**_____.

1.c. 0011_2 je 4-bitni prikaz nekog broja Z u NBC formatu. Broj Z iznosi: ___**3**____. Prikažite broj Z u 6-bitnom prikazu dvojnog komplementa: _____**000011**_____. Prikažite broj Z u 8-bitnom formatu s bitom za predznak_____**00000011**_____.

2. (1 bod) Harvardska arhitektura ima razdvojenu _____**podatkovnu memoriju (ili sabirnicu)**_____ i _____**programsku memoriju (ili sabirnicu)**____. Glavna prednost Harvardske (u odnosu na Von Neumannovu arhitekturu) je _____**veća brzina**___. Prednost Von Neumannove arhitekture je _____**jednostavnost (ili cijena)**_____.

3. (3 boda) Četiri načina rada sklopa FRISC-PIO su: _____**ulazni**_____, _____**izlazni** _____, ____**postavljanje bitova**_____ i _____**ispitivanje bitova**_____. PIO se na vanjski proces spaja pomoću priključaka _____**PIOD**_____ širine ___**8**___ bita te pomoću dva jednobitna priključka _____**READY**_____ i _____**STROBE**_____ koji služe za _____**rukovanje (ili sinkronizaciju)**_____. Ova dva priključka koriste se u načinima rada _____**ulazni**_____ i _____**izlazni**_____. PIO **ne može** postati spreman kad radi u načinu rada _____**postavljanja bitova**_____.

4. (1,5 boda) Potprogram se kod FRISC-a poziva naredbom ____**CALL**___, a kod ARM-a naredbom ____**BL**_____. Povratna adresa se kod FRISC-a sprema _____**na stog**_____ (gdje), a kod ARM-a _____**u LR (ili R14)**_____. Povratak iz potprograma se kod FRISC-a ostvaruje naredbom _____**RET**_____, a kod ARM-a naredbom _____**MOV PC,LR (ili MOV PC,R14)**_____.

5.a. (1,75 boda) Vrste prekida kod FRISC-a su _____**maskirajući**_____(A) i _____**nemaskirajući**_____(B). Adresa prekidnog potprograma (A) je _____**zapisana u memoriji na adresi (lokaciji) 8**_____, a za prekid (B) adresa potprograma je _____**12(dekadski)**_____. Za oba prekida, povratna adresa se sprema _____**na stog**_____(gdje). Povratak iz prekidnog potprograma (A) ostvaruje se naredbom _____**RETI**_____, a iz (B) pomoću naredbe _____**RETN**_____.

5.b. (2,25 boda) Kod ARM-a imamo prekide _____**IRQ (ili obični prekid)**_____(C) i _____**FIQ (ili brzi prekid)**_____(D). Adresa prekidnog potprograma (C) je _____**18(heksa)**_____, a za prekid (D) adresa potprograma je _____**1C(heksa)**_____. Za prekid (C) povratna adresa se sprema _____**u LR_irq**_____(gdje), a za prekid (D) povratna adresa se sprema _____**u LR_fiq**_____(gdje). Povratak iz prekidnog potprograma (C) ostvaruje se naredbom _____**SUBS,PC,LR,#4**_____, a iz (D) pomoću naredbe _____**SUBS,PC,LR,#4**_____. ARM kod prihvaćanja prekida, osim povratne adrese, automatski sprema još i registar ____**CPSR**_____.

6. (2 boda) Na sabirnici AHB čitanje iz **brze** memorije podijeljeno je na _____**adresnu**_____ fazu koja traje __**1**___ takta clock-a i na _____**podatkovnu**_____ fazu koja traje ___**1**___ takta clock-a. Zbog preklapanja ovih faza, efektivno će **četiri** uzastopna brza čitanja trajati ___**5**__ taktova clock-a. Podatkovni priključci procesora ARM ukupno su široki ___**64**____ bita, od čega jedna polovica ima _____**ulazni**_____ smjer, a druga polovica _____**izlazni**_____ smjer.

7. (1 bod) Neka priručna memorija ima 48_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
0x23	0x23
0x48	0x18
0x59	0x29
0x90	0x00

8. (5,5 bodova)

```
DMASRC      `EQU  0FFFF1000
DMADEST     `EQU 0FFFF1004
DMACNT      `EQU  0FFFF1008
DMACTRL     `EQU 0FFFF100C
DMASTART    `EQU 0FFFF1010
DMAIACK     `EQU 0FFFF1014
BVJ         `EQU  0FFFF2000
UVJ1_DATA   `EQU  0FFFF3000
UVJ1_TEST   `EQU  0FFFF3004
UVJ2_DATA   `EQU  0FFFF4000
UVJ2_TEST   `EQU  0FFFF4004

      `ORG 0
      MOVE 10000, SP
GLAVNI
      MOVE 0, R1              ; R1 - brojač podataka UVJ1
      MOVE 0, R2              ; R2 - brojač podataka UVJ2
      MOVE 0, R3              ; R3 - ukupni brojač podataka; može i R1+R2
      MOVE PODACI1, R4
      MOVE PODACI2, R5

C1    LOAD R0, (UVJ1_TEST)    ; prozivanje
      OR R0, R0, R0
      JR_NZ P1
C2    LOAD R0, (UVJ2_TEST)
      OR R0, R0, R0
      JR_Z C1

P2    LOAD R0, (UVJ2_DATA)
      STORE R0, (UVJ2_TEST
      STORE R0, (R5)          ; spremam podatak
      ADD R5, 4, R5           ; pomičem brojač
      ADD R2, 1, R2           ; dodajem 1 za brojač
      ADD R3, 1, R3           ; dodajem 1 za ukupni brojač
      CMP R3, %D 99
      JR_NE C1                ; da ne dođe do izgladnjivanja
      JR_EQ VAN

P1    LOAD R0, (UVJ1_DATA)
      STORE R0, (UVJ1_TEST)
      STORE R0, (R4)          ; spremam podatak
      ADD R4, 4, R4           ; pomičem brojač
      ADD R1, 1, R1           ; dodajem 1 za brojač
      ADD R3, 1, R3           ; dodajem 1 za ukupni brojač
      CMP R3, %D 99
      JR_NE C2

VAN   CMP R1, R2
      JR_P VECI1
      JR_N VECI2
VECI1 MOVE PODACI1, R4
      PUSH R4                 ; adresa početka podataka
      PUSH R1                 ; broj podataka
      JP POZOVI
VECI2 MOVE PODACI2, R5
      PUSH R5                 ; adresa početka podataka
      PUSH R2
POZOVI CALL SALJI
      ADD SP, 8, SP
      HALT

PODACI1 `DS 400
PODACI2 `DS 400
```

```
SALJI
   PUSH R0              ; spremanje na stog
   LOAD R0, (SP+C)      ; učitaj adresu početka bloka
   STORE R0, (DMASRC)   ; izvor: adresa početka bloka
   LOAD R0, (SP+8)      ; učitaj broj podataka
   STORE R0, (DMACNT)   ; broj podataka
   MOVE BVJ, R0
   STORE R0, (DMADEST)  ; adresa BVJ je odredište
   MOVE %B 1000, R0     ; DMA upravljačka riječ
   STORE R0, (DMACTRL)
   STORE R0, (DMASTART
   POP R0
   RET
```

9. (4 boda)

```
        `ORG 0

        MOV SP, #10<8               ; inicijalizacija stoga
        BL PAROVI                   ; poziv 1. potprograma
        HALT

PAROVI
        STMFD SP!, {R0, R1, R2, R3, R4, R14}   ; obavezno i R14 zbog gniježđenja potp.
        LDR R1, POCETAK             ; R1 - adresa početka niza
        LDR R2, BROJ                ; R2 - broj parova
PETLJA
        LDRSB R3, [R1]              ; učitavam prvi 8-bitni broj i proširujem
        LDRSB R4, [R1, #1]          ; učitavam drugi 8-bitni broj i proširujem

        CMP R3, R4                  ; koji je veći?
        STMGEFD SP!, {R3}           ; R3 je veći
        STMLTFD SP!, {R4}           ; R4 je veći

        BL ABS                      ; rezultat je u R0
        ADD SP, SP, #4              ; brišem parametar

        STRH R0, [R1], #2           ; spremam rezultat na iste adrese i povećavam za 2B
        SUB R2, R2, #1              ; smanjujem brojač parova

        CMP R2, #0                  ; jesmo li došli do kraja
        BNE PETLJA

        LDMFD SP!, {R0, R1, R2, R3, R4, R14}
        MOV PC, LR

ABS
        LDR R0, [SP]                ; čitam parametar sa stoga, uklanja ga pozivatelj
        CMP R0, #0                  ; test za postavljanje zastavica
                                    ; pozitivni brojevi su zapravo već OK
        MVNMI R0, R0                ; za negativne brojeve radim XOR
        ADDMI R0, R0, #1            ; i još dodam 1 za 2'k - apsolutna vrijednost
                                    ; ili RSBMI R0, R0, #0 ili MOVE RX, #0 + SUB R0, RX,R0

        MOV PC, LR                  ; sad je u R0 rezultat

POCETAK DW 100<4
BROJ DW 20
```

10. (6 bodova)

```
       `ORG 0
       MOV SP, #10<12            ; init stoga
       B GLAVNI

       `ORG 1C                   ; FIQ
       LDR R8, GPIO              ; pristojno je ponovno učitati adrese, a ne slati ih
       LDR R9, RTC               ; preko registara
       MOV R10, #0               ; R8-R12 su maskirani, ne treba ih čuvati
                                 ; ako se koriste R0-R7, čuvati kontekst!
       STR R10, [R9, #8]         ; prihvaćamo prekid RTC
       STR R10, [R8, #4]         ; isključujemo pećnicu
       SUBS PC, LR, #4           ; izlazimo iz prekidnog potprograma

GLAVNI

       MRS R0, CPSR
       BIC R0, R0, #40           ; omogući FIQ (#40)
       MSR CPSR_c, R0

       LDR R0, GPIO              ; GPIO init
       LDR R6, RTC
       MOV R1, #%B 11111001      ; ulazni + 2 izlazna + ulazni
       STR R1, [R0, #C]          ; spremam u registar smjera B, A je već ulazni

PETLJA                           ; čekalica za signal READY
       LDR R1, [R0, #4]          ; čitam s vrata B
       ANDS R1, R1, #1           ; maskiram za samo zadnji bit
       BEQ PETLJA                ; readyja još nema (bit[0] = 1?] ili CMP pa dalje


       LDR R2 [R0]               ; čitam s vrata A masu i vrstu hrane
       AND R3, R2, #0F           ; R3 - masa hrane
       AND R4, R2, #F0           ; R4 - vrsta hrane
       MOV R4, R4, LSR #4        ; vratim bitove s viših na niže pozicije

       MOV R1, #2                ; šaljem STROBE = 1 (010)
       STR R1, [R0, #4]

       MOV R2, #10<8             ; adresa 1000 - početak tablice
       LDRB R5, [R2, R4]         ; uzimam specifično trajanje iz točnog retka

       LDR R7, SESTO             ; 600 je preveliko za upisivanje
       MUL R5, R5, R7            ; specifično trajanje u sekundama
                                 ; * 60 sekunde, *10 za RTC konstantu = 600 na SESTO
       MUL R5, R5, R3            ; množim s masom hrane
                                 ; u MUL ne smije stajati immediate!!!

       STR R5, [R6, #4]          ; stavljamo konstantu u RTCMR
       MOV R5, #1
       STR R5, [R6, #10]         ; omogućujemo RTC-prekid

       MOV R5, #0
       STR R5, [R6, #C]          ; resetiramo brojilo

       MOV R5, #4                ; %B 100, uključujem pećnicu i STROBE = 0
       STR R5, [R0, #4]          ; može i odvojeni STROBE gore

       B PETLJA

SESTO DW %D 600
GPIO   DW 0FFFF1000
RTC    DW 0FFFF2000
```

Završni ispit iz Arhitekture računala 1 **15. lipnja 2012.** **Grupa na predavanjima: _____**

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 (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Završni ispit traje 150 minuta.

1.a. FRISC (1 bod) Koji će biti sadržaj registra R0, nakon izvođenja naredbe LOAD na FRISC-u za programski odsječak s desne strane: _____

```
`ORG 130
DH 34, 56
DW 8558
...
LOAD  R0, (130)
```

1.b. FRISC (2,5 boda) Prilikom dekodiranja i izvođenja FRISC-ove naredbe **SUB R1, %D10, R2**, broj 10 se nalazi u 20 nižih bitova registra _____. Naredba koristi dva adresiranja koja se zovu _____ i _____. Prilikom izvođenja naredbe, na jedan ulaz ALU dovodi se podatak iz sklopa _____**,** a na drugi se ulaz dovodi podatak iz _____ (*napisati iz kojih dijelova procesora se dovodi podatak*).

1.c. FRISC (2 boda) Prekidni sustav procesora FRISC sastoji se od priključaka: _____. Obzirom na mogućnost programske zabrane/dozvoljavanja prekida, procesor FRISC podržava _____ prekide na priključku/priključcima _____ , te _____ prekide na priključku/priključcima _____. U registru SR nalaze se prekidne zastavice: _____. Prilikom prihvaćanja prekida, procesor FRISC automatski sprema povratnu adresu (*gdje*): _____, a stanje registra SR sprema (*gdje*): _____.

1.d. FRISC (2 boda) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-PIO zovu se: _____ i _____. Ovi priključci koriste se u načinima rada (*nabrojite kojim*): _____. FRISC-PIO **ne može** postati spreman u načinu/načinima rada: _____.

Osim registra maske, na prvoj adresi sklopa PIO nalaze se upravljački registri: _____. Kad šaljemo upravljačku riječ na tu adresu, PIO zna u koji registar je želimo upisati na temelju (*čega?*): _____. Ako želimo da PIO (spojen na adresi FFFF4000) generira prekid kad se na bilo kojem od 3 najniža bita postave nule, onda ga inicijaliziramo tako da pošaljemo podatak _____ na adresu _____ , a zatim podatak _____ na adresu _____.

2.a. ARM (**0,5 boda)** Procesor ARM izvodi odsječak programa s desne strane. Nakon izvođenja naredbe LDRB sadržaj registra R1 će biti: _____, a sadržaj registra R0: _____.

```
`ORG 0
MOV R0, #1<8
LDRB R1, [R0,#4]!
`ORG 100
DW 88776655, 44332211
```

2.b. ARM (0,5 boda) Procesor ARM izvodi naredbu LDMFD R13!, {R2,R1}. Ako je sadržaj registra R13 prije izvođenja naredbe bio 100_{10}, sadržaj registra R13 nakon naredbe će biti _____ , a registar R1 će se napuniti podatkom s memorijske lokacije _____.

2.c. ARM (1 bod) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: _____ , _____ i _____. Uvjetno izvođenje kod procesora ARM moguće je za (*koje?*) _____ naredbe.

2.d. ARM (**0,5 boda**) Za procesor ARM kod naredaba skokova moguća je pojava _____ hazarda. Negativni efekti ovog hazarda se u nekim procesorima umanjuju pomoću postupka _____ .

2.e. ARM (**1 bod**) Procesor ARM9 uvodi _____ arhitekturu memorijskog pristupa. Time se izbjegava _____ hazard koji postoji kod ARM7. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave _____ hazarda čiji se negativni efekti umanjuju pomoću postupka _____.

OKRENUTI STRANICU! ☺

2.f. ARM (**1 bod)** Nakon uključenja procesor ARM 7 treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:_____ (*napisati postupak rješavanja!*)

```
    `ORG 0
    MOV R1, #5
    ADD R0,R1,#2
    BIC R0, R0, #%b100
A   SUBS  R0,R0,#1
    BNE A
    EOR R0,R0,R0
```

2.g. ARM (*2 boda*) Kod ARM-a imamo prekide _____(A) i _____(B). Adresa prekidnog potprograma (A) je _____, a za prekid (B) adresa potprograma je _____. Za prekid (A) povratna adresa se sprema (*gdje*): _____, a za prekid (B) povratna adresa se sprema (*gdje*): _____. Povratak iz prekidnog potprograma (A) ostvaruje se naredbom _____, a iz (B) pomoću naredbe _____.

2.h. ARM (1 bod) Neka priručna memorija ima 16_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
0x10	
0x11	
0x25	
0x3F	

3. (*7 bodova*) U računalnom sustavu nalaze se procesor FRISC, DMA, CT (na ulaz CNT spojen je signal frekvencije 1 kHz) te dva sklopa PIO. Adrese vanjskih jedinica odaberite sami.

Procesor u glavnom programu treba neprestano čitati 8-bitne NBC podatke sa sklopa PIO1 (koji radi u uvjetnom načinu rada). Svaki podatak treba pomnožiti sa 3 i kopirati u sve lokacije odredišnog bloka memorije. Odredišni blokovi počinju od adrese 3000_{16} i nalaze se jedan iza drugoga. Svaki odredišni blok treba sadržavati 20_{16} 32-bitnih podataka. Ako se sa PIO1 učita broj FF_{16}, potrebno je zaustaviti brojenje CT-a te zaustaviti procesor.

Kopiranje vrijednosti u odredišni blok obavlja se pomoću sklopa DMA, koji radi u uvjetnom načinu rada, krađom ciklusa.

Svakih 10 sekundi na sklop PIO2 (koji radi u bezuvjetnom načinu rada) potrebno je poslati ukupni broj pokretanja sklopa DMA. Kašnjenje ostvariti CT-om (spojen na INT2).

4. *(7 bodova*) Za procesor ARM napišite potprogram SAVRSEN koji provjerava je li broj savršen – jednak zbroju svojih pravih djelitelja (tj. zbroju svih svojih djelitelja, uključujući i 1, osim samoga sebe). Na primjer, 28 je savršen jer vrijedi 28=1+2+4+7+14, a 8 nije savršen jer vrijedi 8≠1+2+4. Parametar se u potprogram šalje stogom (uklanja ga pozivatelj), a rezultat se vraća pomoću R0. Rezultat iznosi 1 ako je broj savršen, a 0 ako broj nije savršen.

U glavnom programu treba provjeriti 32-bitne NBC-brojeve iz bloka memorije na adresi 500_{16} i zamijeniti nulom one koji nisu savršeni. U bloku je 10_{10} podataka.

Potprogram SAVRSEN mora djeljivost brojeva provjeravati pomoću potprograma DJELJIVO. Napišite potprogram DJELJIVO koji provjerava je li prvi parametar (prenosi se registrom R1) djeljiv s drugim parametrom (prenosi se lokacijom iza naredbe poziva potprograma). Ako je djeljiv, preko R0 se vraća broj 1, a inače se vraća 0.

5. (*6 bodova*) Računalni sustav inkubatora za patkice sastoji se od procesora ARM te sklopova RTC (spojen na signal FIQ) i GPIO (na koji je spojen termometar opisan na predavanjima).

- vrata B, bitovi [5:0]: iznos temperature, 6-bitni NBC.
- vrata B, bit 6: signal (aktivan visoko) kojim termometar dojavljuje GPIO-u da je postavljena nova temperatura
  vrata B, bit 7: signal (aktivan visoko) kojim GPIO dojavljuje termometru da je pročitao temperaturu
- vrata A, bit 0: izlazni bit kojim se uključuje grijalica (1 uključuje, a 0 isključuje)
- vrata A, bit 1: izlazni bit kojim se uključuje hladilica (1 uključuje, a 0 isključuje)

Napišite program koji upravlja radom inkubatora i treba održavati željenu temperaturu. Temperaturu treba regulirati svakih 60 sekundi (na ulaz RTC-a spojen je signal od 1 Hz). Na početku željena temperatura treba biti 40 stupnjeva. Svaki treći dan (3 dana = 4320 minuta) treba smanjivati željenu temperaturu za 1 stupanj. Nakon 30 dana, treba zaustaviti rad programa.

Završni ispit iz Arhitekture računala 1

15. lipnja 2012. **Grupa na predavanjima: _____**

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 (popis naredaba FRISC-a i ARM-a). Programe treba pisati uredno i komentirati pojedine cjeline programa. Sve teorijske zadatke rješavati na ovaj papir. Završni ispit traje 150 minuta.

1.a. FRISC (1 bod) Koji će biti sadržaj registra R0, nakon izvođenja naredbe LOAD na FRISC-u za programski odsječak s desne strane: _____**00560034**_____

```
`ORG 130
DH 34, 56
DW 8558
...
LOAD  R0, (130)
```

1.b. FRISC (2,5 boda) Prilikom dekodiranja i izvođenja FRISC-ove naredbe **SUB R1, %D10, R2**, broj 10 se nalazi u 20 nižih bitova registra **____IR_____**. Naredba koristi dva adresiranja koja se zovu **__registarsko__** i **____neposredno/immediate_____**. Prilikom izvođenja naredbe, na jedan ulaz ALU dovodi se podatak iz sklopa **____EXT____(nije bitan redoslijed EXT i R1)__,** a na drugi se ulaz dovodi podatak iz **__R1___** (*napisati iz kojih dijelova procesora se dovodi podatak*).

1.c. FRISC (2 boda) Prekidni sustav procesora FRISC sastoji se od priključaka: **____INT0-INT3, IACK____**. Obzirom na mogućnost programske zabrane/dozvoljavanja prekida, procesor FRISC podržava **__maskirajuće__** prekide na priključku/priključcima **__INT0-INT2__**, i **___nemaskirajuće** ___ prekide na priključku/priključcima **__INT3___**. U registru SR nalaze se prekidne zastavice: ___ **EINT0-EINT2, GIE___**. Prilikom prihvaćanja prekida, procesor FRISC automatski sprema povratnu adresu (*gdje*): **____na stog__**, a stanje registra SR sprema (*gdje*): **___ne sprema se nigdje__**.

1.d. FRISC (2 boda) Priključci za rukovanje (sinkronizaciju) kod sklopa FRISC-PIO zovu se: **____READY ____ i ____ STROBE___.** Ovi priključci koriste se u načinima rada (*nabrojite kojim*): **___ulazni, izlazni___.** FRISC-PIO **ne može** postati spreman u načinu/načinima rada: _____**postavljanje bitova**____.

Osim registra maske, na prvoj adresi sklopa PIO nalaze se upravljački registri: **__ICR i OCR_**. Kad šaljemo upravljačku riječ na tu adresu, PIO zna u koji registar je želimo upisati na temelju (*čega?*): **__najnižeg bita poslane upravljačke riječi (ili poslanog podatka)** _____. Ako želimo da PIO (spojen na adresi FFFF4000) generira prekid kad se na bilo kojem od 3 najniža bita postave nule, onda ga inicijaliziramo tako da pošaljemo podatak **__00001111_2___** na adresu _ **FFFF4000**__i zatim podatak _____**00000111_2**_____ na adresu __ **FFFF4000**___.

2.a. ARM (**0,5 boda**) Procesor ARM izvodi odsječak programa s desne strane. Nakon izvođenja naredbe LDRB sadržaj registra R1 će biti: ___**0x11**___, a sadržaj registra R0: ___**0x104**___.

```
`ORG 0
MOV R0, #1<8
LDRB R1, [R0,#4]!
`ORG 100
DW 88776655, 44332211
```

2.b. ARM (0,5 boda) Procesor ARM izvodi naredbu LDMFD R13!, {R2,R1}. Ako je sadržaj registra R13 prije izvođenja naredbe bio 100_{10}, sadržaj registra R13 nakon naredbe će biti **__108_{10}__** , a registar R1 će se napuniti podatkom s memorijske lokacije **__100_{10}__**.

2.c. ARM (1 bod) Kod procesora ARM7 postoje tri različite grupe naredaba s obzirom na način kako se naredbe izvode. To su naredbe za: **___obradu podataka / AL naredbe__**, **__prijenos podataka / load-store / memorijske___** i **____grananje / upravljačke___**. Uvjetno izvođenje kod procesora ARM moguće je za (*koje?*) **___ sve / gotovo sve___** naredbe.

2.d. ARM (**0,5 boda**) Za procesor ARM kod naredaba skokova moguća je pojava _____**upravljačkog**_____ hazarda. Negativni efekti ovog hazarda se u nekim procesorima umanjuju pomoću postupka **___predviđanja grananja**_____ .

2.e. ARM (**1 bod**) Procesor ARM9 uvodi **__harvardsku**____ arhitekturu memorijskog pristupa. Time se izbjegava **__strukturni___** hazard koji postoji kod ARM7. Međutim, zbog razdvajanja razine izvođenja na 3 nove protočne razine dolazi do mogućnosti pojave **__podatkovnog__** hazarda čiji se negativni efekti umanjuju upotrebom **___prosljeđivanja rezultata/result forwarding__**.

2.f. ARM (**1 bod)** Nakon uključenja procesor ARM 7 treba izvesti programski odsječak s desne strane. Koliko vremenskih perioda traje izvođenje ovog programskog odsječka:_____**16**____. (*napisati postupak rješavanja!*)

```
    `ORG 0
    MOV R1, #5
    ADD R0,R1,#2
    BIC R0, R0, #%b100
A   SUBS  R0,R0,#1
    BNE A
    EOR R0,R0,R0
```

2.g. ARM (*2 boda*) Kod ARM-a imamo prekide **_IRQ (ili obični prekid)_**(A) i **___FIQ (ili brzi prekid)___**(B). Adresa prekidnog potprograma (A) je **__0x18__**, a za prekid (B) adresa potprograma je **__0x1C___**. Za prekid (A) povratna adresa se sprema (*gdje*): **__u LR_irq__**, a za prekid (B) povratna adresa se sprema (*gdje*): **__u LR_fiq__**. Povratak iz prekidnog potprograma (A) ostvaruje se naredbom **__SUBS,PC,LR,#4__**, a iz (B) pomoću naredbe **__SUBS,PC,LR,#4__**.

2.h. ARM (1 bod) Neka priručna memorija ima 16_{10} blokova i direktno preslikavanje. Za zadane adrese blokova u radnoj memoriji (zadane su heksadekadski) odredite u kojem bloku priručne memorije će se nalaziti:

Adresa bloka u radnoj memoriji	Adresa bloka u priručnoj memoriji
0x10	**0x0**
0x11	**0x1**
0x25	**0x5**
0x3F	**0xF (ili 15)**

3. (*7 bodova*) U računalnom sustavu nalaze se procesor FRISC, DMA, CT (na ulaz CNT spojen je signal frekvencije 1 kHz) te dva sklopa PIO. Adrese vanjskih jedinica odaberite sami.

Procesor u glavnom programu treba neprestano čitati 8-bitne NBC podatke sa sklopa PIO1 (koji radi u uvjetnom načinu rada). Svaki podatak treba pomnožiti sa 3 i kopirati u sve lokacije odredišnog bloka memorije. Odredišni blokovi počinju od adrese 3000_{16} i nalaze se jedan iza drugoga. Svaki odredišni blok treba sadržavati 20_{16} 32-bitnih podataka. Ako se sa PIO1 učita broj FF_{16}, potrebno je zaustaviti brojenje CT-a te zaustaviti procesor.

Kopiranje vrijednosti u odredišni blok obavlja se pomoću sklopa DMA, koji radi u uvjetnom načinu rada, krađom ciklusa.

Svakih 10 sekundi na sklop PIO2 (koji radi u bezuvjetnom načinu rada) potrebno je poslati ukupni broj pokretanja sklopa DMA. Kašnjenje ostvariti CT-om (spojen na INT2).

```
PIO1  `EQU 0FFFF1000
PIO2  `EQU 0FFFF2000
DMA   `EQU 0FFFF3000
CT    `EQU 0FFFF4000

`ORG 0
      MOVE 10000, SP            ; inicijalizacija stoga
      JP MAIN                   ; skok u glavni program
`ORG 8
      DW 2000                   ; prekidni vektor, običan prekid

GLAVNI
      MOVE 3000, R1             ; R1 – adresa odredišnog bloka

      MOVE %D 10000, R0         ; 10 kHz * 10 sekundi, CT-konstanta
      STORE R0, (CT)
      MOVE %B11, R0             ; CR, brojilo broji, prekid
      STORE R0, (CT+4)
      MOVE %B 11000000, SR      ; omogućavanje prekida, INT2

      MOVE %B 000001, R0        ; PIO1 – ulazni, bez prekida, ICR
      STORE R0, (PIO1)

      MOVE %B 100, R0           ; PIO2 – postav. bitova, bez prekida, OCR
      STORE R0, (PIO2)
```

```
 PETLJA LOAD R0, (PIO1)         ; čekanje na spremnost PIO1
      CMP R0, 0
      JP_EQ PETLJA
      LOAD R0, (PIO1+4)         ; učitavanje podatka s PIO1
      CMP R0, 0FF               ; ako je podatak FF -> kraj
      JP_EQ KRAJ

      SHL R0, 1, R2             ; množenje s 3 = množenje s 2 +...
      ADD R2, R0, R0            ; ...+ zbrajanje
      STORE R0, (POMOC)         ; broj treba spremiti u memoriju zbog DMA
      STORE R0, (PIO1+8)        ; brisanje spremnosti PIO1
      STORE R0, (PIO1+0C)       ; dojava kraja posluživanja PIO1

      MOVE POMOC, R0            ; adresu POMOC postaviti kao izvor DMA
      STORE R0, (DMA+0)
      STORE R1, (DMA+4)         ; adresa trenutnog odredišnog bloka
      ; Update adrese odredišnog bloka (gornja naredba) nije potreban, jer će
      ; adresni registar u DMA-sklopu tijekom DMA-prijenosa biti točno povećan
      MOVE 20, R0               ; brojač podataka u bloku
      STORE R0, (DMA+8)
      MOVE %B 0110, R0          ; kopiranje! SRC-vanjska, DST-memorija
      STORE R0, (DMA+0C)        ; krađa ciklusa, bez prekida
      STORE R0, (DMA+10)        ; pokretanje DMA
DMAW  LOAD R0, (DMA+0C)         ; čekanje na spremnost DMA
      CMP R0,0
      JP_EQ DMAW

      LOAD R0, (BROJ)           ; brojač DMA-prijenosa
      ADD R0, 1, R0
      STORE R0, (BROJ)

      STORE R0, (DMA+14)        ; brisanje DMA-status bistabila
      ADD R1, 80, R1            ; povećavanje odredišnog bloka za 4*20=80(16)

      JP PETLJA                 ; skok na novo čekanje
KRAJ  MOVE 0, R0                ; zaustavljanje CT-a
      STORE R0, (CT+4)
      HALT

      `ORG 2000                 ; prekidni potprogram
INT   PUSH R0                   ; čuvanje konteksta
      MOVE SR, R0
      PUSH R0

      STORE R0, (CT+8)          ; potvrda prihvata prekida
      LOAD R0, (BROJ)           ; učitavanje broja pokretanja DMA
      STORE R0, (PIO2+4)        ; slanje broja na PIO2, bezuvjetno!
      STORE R0, (CT+0C)         ; kraj posluživanja

      POP R0                    ; obnova konteksta
      MOVE R0, SR
      POP R0
      RETI                      ; povratak iz p.p.

BROJAC DW 0                     ; BROJAC – broj odrađenih DMA-prijenosa
POMOC DW 0                      ; POMOC – mjesto za spremanje broja u mem
```

4. *(7 bodova*) Za procesor ARM napišite potprogram SAVRSEN koji provjerava je li broj savršen – jednak zbroju svojih pravih djelitelja (tj. zbroju svih svojih djelitelja, uključujući i 1, osim samoga sebe). Na primjer, 28 je savršen jer vrijedi 28=1+2+4+7+14, a 8 nije savršen jer vrijedi 8≠1+2+4. Parametar se u potprogram šalje stogom (uklanja ga pozivatelj), a rezultat se vraća pomoću R0. Rezultat iznosi 1 ako je broj savršen, a 0 ako broj nije savršen.

U glavnom programu treba provjeriti 32-bitne NBC-brojeve iz bloka memorije na adresi 500_{16} i zamijeniti nulom one koji nisu savršeni. U bloku je 10_{10} podataka.

Potprogram SAVRSEN mora djeljivost brojeva provjeravati pomoću potprograma DJELJIVO. Napišite potprogram DJELJIVO koji provjerava je li prvi parametar (prenosi se registrom R1) djeljiv s drugim parametrom (prenosi se lokacijom iza naredbe poziva potprograma). Ako je djeljiv, preko R0 se vraća broj 1, a inače se vraća 0.

```
GL    MOV SP, #1<16              ; inicijalizacija stoga
      MOV R6, #5<8               ; R6 – adresa početka bloka (500)
      MOV R2, #0A                ; R2 – broj brojeva u bloku (10)

POC   LDR R0, [R6]               ; učitavanje podatka
      STMFD R13!, {R0}           ; stavljanje parametra na stog
      BL SAVRSEN                 ; poziv potprograma SAVRSEN
      ADD R13, R13, #4           ; brisanje parametra sa stoga
      CMP R0, #0                 ; je li broj NEsavršen?
      STREQ R0, [R6]             ; upisivanje 0 umjesto nesavršenog broja
      ADD R6, R6, #4             ; sljedeći podatak
      SUBS R2, R2, #1            ; smanjivanje broja podataka
      BNE POC                    ; natrag na početak petlje
KRAJ  HALT


SAVRSEN
      STMFD R13!, {R1,R3,R4,R5,R14}   ; kontekst, R14 - gniježđenje
      ADD R3, R13, #14           ; „preskakanje“ konteksta, dolazak do parametra
      LDMFD R1, {R3}             ; R3 - parametar (broj) ili LDR R1,[R13,#14]
      MOV R4, #0                 ; R4 - zbroj djelitelja
      MOV R5, #0                 ; R5 – mogući djelitelji, od (1) do R3
TESTIRAJ
      ADD R5, R5, #1             ; uvećavanje djelitelja
      CMP R1, R5                 ; ako je djelitelj = broj -> kraj
      BEQ  GOTOVO                ; došao je do kraja

      STR R5, PARAMETAR          ; parametar ide na adresu iz poziva potp.
      BL   DJELJIVO              ; poziv potprograma
PARAMETAR  DW 0                  ; parametar je odmah iza potprograma
      CMP  R0, #1                ; R0 – rezultat. Djeljivost? 1 – da
      ADDEQ R4,R4,R5             ; djeljivo – dodavanje djelitelj u zbroj
      B TESTIRAJ                 ; sljedeći djelitelj
GOTOVO
      CMP R1, R4                 ; usporedba zbroj = početni broj?
      MOVNE R0, #0               ; vraćanje 0 - NEsavršen
      MOVEQ R0, #1               ; vraćanje 1 - savršen
      LDMFD R13!, {R1,R3,R4,R5,R14}; vraćanje konteksta, R14!!!
      MOV PC, LR                 ; povratak iz potprograma SAVRSEN


DJELJIVO
      STMFD R13!, {R1,R2}        ; spremanje konteksta
      MOV R0, #0                 ; priprema povratne vrijednosti false (0)
                                 ; R1 - prvi parametar
      LDR R2, [LR],#4            ; R2 - drugi parametar i istovremeno
                                 ; povećavanje LR (može i posebnom naredbom)
PONOVO SUBS R1,R1,R2             ; uzastopno oduzimanje
      MOVEQ R0, #1
      BHI PONOVO
```

```
      MOVEQ R0, #1              ; vraćanje true (1) ako je djeljiv

      LDMFD R13!, {R1,R2}
      MOV PC, LR
`ORG 500
      DW 1,2,8,%D28,5,6,7,8,9,0A
```

5. (*6 bodova*) Računalni sustav inkubatora za patkice sastoji se od procesora ARM te sklopova RTC (spojen na signal FIQ) i GPIO (na koji je spojen termometar opisan na predavanjima).

- vrata B, bitovi [5:0]: iznos temperature, 6-bitni NBC.
- vrata B, bit 6: signal (aktivan visoko) kojim termometar dojavljuje GPIO-u da je postavljena nova temperatura
  vrata B, bit 7: signal (aktivan visoko) kojim GPIO dojavljuje termometru da je pročitao temperaturu
- vrata A, bit 0: izlazni bit kojim se uključuje grijalica (1 uključuje, a 0 isključuje)
- vrata A, bit 1: izlazni bit kojim se uključuje hladilica (1 uključuje, a 0 isključuje)

Napišite program koji upravlja radom inkubatora i treba održavati željenu temperaturu. Temperaturu treba regulirati svakih 60 sekundi (na ulaz RTC-a spojen je signal od 1 Hz). Na početku željena temperatura treba biti 40 stupnjeva. Svaki treći dan (3 dana = 4320 minuta) treba smanjivati željenu temperaturu za 1 stupanj. Nakon 30 dana, treba zaustaviti rad programa.

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

      `ORG 1C                   ; prekidni potprogram - FIQ

      STMFD R13!,{R0,R1,R4,R5,R6,R7} ; spremanje konteksta za opće registre
      LDR R0, GPIO              ; pristojno je ponovno učitati bazne adrese,
      LDR R1, RTC               ; a ne prenositi ih registrima u p.p.

      MOV R9, #0                ;
      STR R9, [R1,#0C]          ; resetiranje brojača
      STR R9, [R1,#8]           ; dojava prihvata prekida

ČITAJ LDR R10, [R0,#4]          ; čitanje podatka s vrata B
      ANDS R12, R10, #40        ; 40 = %B 100 0000; bit 6 je signal nove temp.
      BEQ ČITAJ                 ; čekanje na novu temperaturu

      MOV R12, #80              ; 80 = %B 1000 0000; bit 7 u visoko
      STR R12, [R0,#4]          ; slanje na vrata B
      MOV R12, #00              ; 80 = %B 0000 0000; bit 7 natrag u nisko
      STR R12, [R0,#4]          ; slanje na vrata B

      AND R10, R10, #3F         ; 3F = 0011 1111; čitanje temperature [5:0]
      LDR R4, ZELJENA           ; učitavanje željene temperature iz mem
      CMP R10, R4               ; usporedba sa željenom temperaturom
      MOVHI R10, #2             ; veća temp – hladilica! = 10
      MOVMI R10, #1             ; manja temp – grijalica! = 01
      MOVEQ R10, #0             ; ista temperatura – isključiti sve = 00
      STR R10, [R0]             ; spremanje na vrata A

      LDR R5, MINUTEU3          ; učitavanje trenutno proteklih minuta
      ADD R5, R5, #1            ; prošla je jedna minuta (zatražen prekid)
      LDR R7, MIN4320           ; učitavanje broja minuta u 3 dana
      CMP R5, R7                ; jesu li prošla 3 dana?
      BNE  NISU3                ; ako nisu prosla 3 dana -> preskoči

JESU3 SUB R4, R4, #1            ; prošla 3 dana -> smanjiti željenu temp. za 1
      MOV R5, #0                ; i vratiti brojač minuta na 0
```

```
      STR R4, ZELJENA           ; spremanje željene temperature u mem

      LDR R6, 3DANA             ; učitavanje trenutno prošlih perioda od 3 dana
      ADD R6, R6, #1            ; 3 dana su prošla, a...
      CMP R6, #%D 10            ; ...ukupno 30 dana rada inkubatora = 3*10
      STR R6, 3DANA             ; spremanje trenutno prošlih perioda od 3 dana
      BNE  NISU3                ; ako nije prošlo 30 dana -> preskoči kraj
      SWI 123456                ; kraj

NISU3 STR R5, MINUTEU3          ; spremanje trenutno proteklih minuta
      LDMFD R13!,{R0,R1,R4,R5,R6,R7} ; vraćanje konteksta
      SUBS PC, LR, #4           ; povratak iz prekidnog potprograma

GLAVNI
      MOV R13,#10<12            ; inicijalizacija stoga
      LDR R0, GPIO              ; bazna adresa sklopa GPIO
      LDR R1, RTC               ; bazna adresa sklopa RTC

      MOV R2, #3                ; 3 = %B 11, registar smjera vrata A
      STR R2, [R0,#8]           ; bitovi 0 i 1 su izlazni (1)
      MOV R2, #7F               ; 7F = %B 0111 1111, registar smjera vrata B
      STR R2, [R0,#C]           ; bit 7 je izlazni (0), ostali ulazni (1)
      MOV R2, #1                ; 1 - prekidi
      STR R2, [R1,#10]          ; upis u upravljački registar (CTCR)
      MOV R2, #%D60             ; 60 sekundi
      STR R2, [R1,#4]           ; – upis u RTCMR

      MRS R2, CPSR              ; omogućavanje FIQ-prekida
      BIC R2, R2, #40
      MSR CPSR_c, R2
PETLJA    B PETLJA              ; prazna petlja
      `ORG 300
GPIO  DW FFFF1000
RTC   DW FFFF2000
MINUTEU3 DW 0                   ; trenutno prošle minute u 3-dnevnom periodu
3DANA DW 0                      ; trenutno protekli 3-dnevni periodi
ZELJENA DW %D 40                ; željena temperatura, na početku 40
MIN4320 DW %D 4320              ; broj minuta u 3 dana
```