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).** Ne treba popuniti sve crte:

Razina dohvata:

Prva polovina periode CLOCK-a:

_____

Druga polovina periode CLOCK-a:

_____

_____

_____

_____

_____

_____

Razina izvođenja:

Prva polovina periode CLOCK-a:

_____

_____

_____

Druga polovina periode 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                  _____
        ORG 0                  _____
        MOVE   4,  R0          _____
POC     SUB    R0, DVA, R0     _____
        JR_NE  POC             _____
        STORE  R0, (1000)      _____
```

HALT _____

Kraj **svake** naredbe napišite koliko puta se naredba izvodi po **koliko** ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c. 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:

Prva polovina periode CLOCK-a:

____**PC -> AR**_____

Druga polovina periode CLOCK-a:

____ (AR) -> IR_____

_____dekodiranje_____

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

___**ALU: izvodi zbrajanje** _____

____PC +4 -> PC_____

_ **onemogući dohvat u sljedećem ciklusu** _

Razina izvođenja:

Prva polovina periode CLOCK-a:

___ALU -> AR _____

___**R1 -> DR**_____

Druga polovina periode CLOCK-a:

__ **omogući dohvat u sljedećem ciklusu** ___

1 c) (*1,5 boda*) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: **__ADR__**, **__DATA__**, **__RD___**, **____WR____**, **_____SIZE___** i **___WAIT___**.

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                    __0_____________________
        ORG 0                    __0_____________________
        MOVE   4,  R0            __1 x 1c________________
POC     SUB    R0, DVA, R0       __2 x 1c________________
        JR_NE  POC               __1 x 2c + 1 x 1c_______
        STORE  R0, (1000)        __1 x 2c________________
        HALT                     __1 x 2c________________
```

Kraj svake naredbe napišite koliko puta se naredba izvodi po koliko ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c. Izvođenje cijelog programa ukupno traje ___10_____ 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

NEG   AND R2, 7F, R2        ; brisanje predznaka
      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 INT) i PVJ4 (spojena na NMI). 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_IACK  EQU   0FFFF4004
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)
      STORE R0, (PVJ4_IACK)     ; dojava prihvata prekida
      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 10000, SR        ; dozvoli prekide INT

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, može 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
```