

# 1. međuispit iz Arhitekture računala 1

25. travnja 2014.

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). Programme treba pisati uredno i komentirati pojedine cjeline programa. Sve zadatke rješavati na ovaj papir. Međuispit traje 120 minuta.

**1a. (3 boda)** U memoriju čije riječi su široke 8 bita podatci se upisuju u redoslijedu little-endian. Od adrese  $100_{16}$  upišite redom sljedeće podatke (tablicu popunite u heksadekadskoj brojevnoj bazi):

-1030<sub>10</sub> u 32-bitom formatu 2'k.  
1030<sub>10</sub> u 32-bitnom formatu NBC,  
broj 41<sub>10</sub> u 16-bitnom formatu 2'k,  
41<sub>10</sub> u 8-bitnom formatu s bitom za predznak,  
-41<sub>10</sub> u 8-bitnom formatu s bitom za predznak.

100	
101	
102	
103	
104	
105	
106	
107	
108	
109	
10A	
10B	

**MORA SE VIDJETI POSTUPAK PRETVORBE A NE SAMO REZULTATI !!!**

**1b. (2,5 boda)** Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe LOAD R0, (R1+300). 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:

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

**1c. (0,5 boda)** Naredba LOAD efektivno traje \_\_\_\_ ciklusa (uz pretpostavku da je memorija brza). Ovaj hazard naziva se \_\_\_\_\_.

**1d. (1 bod)** Čitanje iz brze memorije traje \_\_\_\_\_ takt(ova) CLOCK-a, a pisanje traje \_\_\_\_\_ takt(ova). Pisanje u sporu memoriju traje \_\_\_\_\_ takt(ova). Ako je memorija spora, to javlja FRISC-u preko sabirničke linije spojene na njegov priključak \_\_\_\_\_, kojega FRISC ispituje \_\_\_\_\_ (u kojem trenutku).

**1e. (1 bod)** Smjerovi FRISC-ovih priključaka su: READ je \_\_\_\_\_, WRITE je \_\_\_\_\_, WAIT je \_\_\_\_\_, ADR je \_\_\_\_\_, DATA je \_\_\_\_\_.

**1f. (1 bod)** Napišite koje se vrste procesorskog adresiranja koriste u sljedećim naredbama:

Naredba MOVE 30, R1 ima dvije vrste adresiranja: \_\_\_\_\_ (30) i \_\_\_\_\_ (R1).

U naredbi JP (R3) koristi se \_\_\_\_\_ adresiranje.

Adresiranje u drugom operandu naredbe LOAD R1,(LABELA) naziva se \_\_\_\_\_.

Adresiranje u drugom operandu naredbe LOAD R1,(SP+4) naziva se \_\_\_\_\_.

**1g. (1 bod)** Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0110 - 1101.

Nakon operacije će biti: prijenos = \_\_\_\_\_, posudba = \_\_\_\_\_, preljev = \_\_\_\_\_, ničtica = \_\_\_\_\_, predznak = \_\_\_\_\_.

**Potrebno je napisati postupak rješenja, a ne samo rezultate.**

## Rješenja zadatka 1a-1g

**1a. (3 boda)** U memoriju čije riječi su široke 8 bita podatci se upisuju u redoslijedu little-endian. Od adrese  $100_{16}$  upišite redom sljedeće podatke (tablicu popunite u **heksadekadskoj** brojevnoj bazi):

- $1030_{10}$  u 32-bitom formatu 2'k.       $FF\ FF\ 11111011\ 11111010 = FF\ FF\ FB\ FA$   
 $1030_{10}$  u 32-bitnom formatu NBC,       $00000100\ 00000110 = 00\ 00\ 04\ 06$   
broj  $41_{10}$  u 16-bitnom formatu 2'k,       $00000000\ 00101001 = 00\ 29$   
 $41_{10}$  u 8-bitnom formatu s bitom za predznak,       $00101001 = 29$   
 $-41_{10}$  u 8-bitnom formatu s bitom za predznak.       $10101001 = A9$

100	FA
101	FB
102	FF
103	FF
104	06
105	04
106	00
107	00
108	29
109	00
10A	29
10B	A9

## **MORA SE VIDJETI POSTUPAK PRETVORBE A NE SAMO REZULTATI !!!**

**1b. (2,5 boda)** Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe LOAD R0, (R1+300). Ne treba popuniti sve crte.

Razina dohvata:

Prva polovina periode CLOCK-a:

PC -> AR

Druga polovina periode CLOCK-a:

(AR) -> IR,

dekodiranje

ext 300 i R1 -> ALU

ALU: izvodi zbrajanje

PC +4 -> PC

onemogućiti dohvat u sljedećem ciklusu

Razina izvođenja:

Prva polovina periode CLOCK-a:

ALU -> AR

R1 -> DR

Druga polovina periode CLOCK-a:

(AR) -> DR

DR -> R0

omogućiti dohvat u sljedećem ciklusu

**1c. (0,5 boda)** Naredba LOAD efektivno traje 2 ciklusa (uz pretpostavku da je memorija brza). Ovaj hazard naziva se strukturni.

**1d. (1 bod)** Čitanje iz brze memorije traje 1 takt(ova) CLOCK-a, a pisanje traje 1 takt(ova). Pisanje u sporu memoriju traje 2 ili više takt(ova). Ako je memorija spora, to javlja FRISC-u preko sabirničke linije spojene na njegov priključak WAIT, kojega FRISC ispituje nakon rastućeg brida signala CLOCK (ili u drugoj polovici periode CLOCK-a) (u kojem trenutku).

**1e. (1 boda)** **Smjerovi** FRISC-ovih priključaka su: RD je izlazni, WR je izlazni, WAIT je ulazni, ADR je izlazni, DATA je ulazno/izlazni (ili dvosmjerni).

**1f. (1 bod)** Napišite koje se vrste **procesorskog adresiranja** koriste u sljedećim naredbama:

Naredba MOVE 30, R1 ima dvije vrste adresiranja: neposredno (30) i registarsko (R1).

U naredbi JP (R3) koristi se registarsko indirektno adresiranje.

Adresiranje u **drugom operandu** naredbe LOAD R1,(LABELA) naziva se apsolutno.

Adresiranje u **drugom operandu** naredbe LOAD R1,(SP+4) naziva se registarsko indirektno s odmakom (ili s offsetom).

**1g. (1 bod)** Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0110 - 1101.

Nakon operacije će biti: prijenos = 0, posudba = 1, preljev = 1, ničtica = 0, predznak = 1.

**Potrebno je napisati postupak rješenja, a ne samo rezultate.**

**2. (6 bodova)** U memoriji na adresama  $A000_{16}$  i  $ABCD1234_{16}$  se nalaze dva bloka sa po  $200_{16}$  podataka u svakom bloku. Prvi blok sadrži 8-bitne podatke u formatu 1'k. Drugi blok sadrži 16-bitne podatke u formatu s bitom za predznak. Treba zbrajati podatke iz prvog bloka s podacima u drugom bloku i spremati rezultate u 32-bitnom formatu 2'k u treći blok memorije na adresi  $C000_{16}$  na sljedeći način: prvi podatak iz prvog bloka zbraja se sa zadnjim podatkom iz drugog bloka, zatim drugi podatak iz prvog bloka s predzadnjim podatkom iz drugog bloka i tako dalje dok se ne zbroje svi podatci.

```

ORG 0

GLAVNI MOVE    0A000, R0          ; adresa 1. bloka
      LOAD     R1, (ADR_2_BLOKA) ; učitaj adresu 2. bloka
      ADD      R1, 400, R1        ; izračunaj adresu zadnjeg podatka u 2. bloku
      SUB      R1, 2, R1
      MOVE     0C000, R2          ; adresa rezultatnog bloka

      MOVE     200, R7            ; brojač za petlju

PETLJA LOADB   R3, (R0)           ; učitaj podatak iz 1. bloka
      AND      R3, 80, R5         ; pretvori ga u 32-bitni 2'k format
      JR_Z     POZ_1             ; ispitaj predznak
NEG_1  SHL      R3, %D24, R3      ; proširi 1'k sa 8 na 32 bita
      ASHR     R3, %D24, R3
      ADD      R3, 1, R3          ; pretvori u 2'k
POZ_1  ; broj je već OK jer je pozitivan

      LOADH    R4, (R1)           ; učitaj podatak iz 2. bloka
      AND      R4, 8000, R5       ; pretvori ga u 32-bitni 2'k format
      JR_Z     POZ_2             ; ispitaj predznak
NEG_2  AND      R4, 0FFFF7FFF, R4 ; briši bit za predznak
      XOR      R4, 0FFFFFFF, R4   ; operacija dvojnog komplementa
      ADD      R4, 1, R4
POZ_2  ; broj je već OK jer je pozitivan

ZBROJI ADD     R3, R4, R5          ; zbroji brojeve u 32-bitnom formatu 2'k
      STORE    R5, (R2)          ; spremi u rezultatni blok

      ADD      R0, 1, R0          ; pomakni pokazivače na blokove
      SUB      R1, 2, R1
      ADD      R2, 4, R2

      SUB      R7, 1, R7          ; smanjivanje i provjera brojača petlje
      JR_NZ    PETLJA

      HALT

ADR_2_BLOKA DW 0ABCD1234          ; adresa drugog bloka

      ORG 0A000                  ; prvi blok
      DB ...
      ORG 0C000                  ; rezultatni blok
      DW ...
      ORG 0ABCD1234              ; drugi blok
      DH ...

```

**3. (7,5 bodova)** Napisati potprogram SUB\_DBL za oduzimanje dva NBC broja u dvostrukoj preciznosti. Parametri i rezultat se prenose registrima na sljedeći način:

–	R1	R0	prvi podatak
	R3	R2	drugi podatak
	R1	R0	rezultat

Napisati potprogram DIV\_DBL koji (koristeći potprogram SUB\_DBL) dijeli dva NBC broja u dvostrukoj preciznosti metodom uzastopnog oduzimanja. Ostatak pri dijeljenju se zanemaruje. Parametri su adrese dvaju podataka i te adrese se šalju pomoću stoga u potprogram. Rezultat dijeljenja vraća se u registrima R4 (niži dio) i R5 (viši dio).

Napisati glavni program koji pozivom potprograma DIV\_DBL dijeli brojeve  $5555666677778888_{16}$  i  $1111222233334444_{16}$  koji su spremjeni u memoriji na adresama  $200_{16}$  i  $208_{16}$  i pohranjuje rezultat na adresu  $210_{16}$  (ovi brojevi i mjesto za rezultat trebaju također biti napisani u rješenju).

ORG 0

GLAVNI MOVE 10000, SP ; inicijalizacija stoga

MOVE 200, R0 ; adresa 1. broja

PUSH R0

MOVE 208, R0 ; adresa 2. broja

PUSH R0

CALL DIV\_DBL ; dijeljenje

ADD SP, 8, SP ; čisti stog od parametara

STORE R4, (210) ; spremi niži dio rezultata

STORE R5, (214) ; spremi viši dio rezultata

HALT

SUB\_DBL ; potprogram za oduzimanje, nema konteksta

SUB R0, R2, R0

SBC R1, R3, R1 ; C je postavljen za ispitivanje u pozivatelju

RET

DIV\_DBL ; potprogram za dijeljenje

PUSH R0 ; spremi kontekst

PUSH R1

PUSH R2

PUSH R3

LOAD R4, (SP+%D20) ; učitaj 2. parametar

LOAD R5, (SP+%D24) ; učitaj 1. parametar

LOAD R0, (R5) ; učitaj prvi broj

LOAD R1, (R5+4)

LOAD R2, (R4) ; učitaj drugi broj

LOAD R3, (R4+4)

MOVE 0, R4 ; rezultat je inicijalno 0

MOVE 0, R5

PETLJA CALL SUB\_DBL

JR\_NC VAN ; ako je negativan broj, dijeljenje je gotovo

POZIT ADD R4, 1, R4 ; ako je pozitivan broj, povećaj rezultat

ADC R5, 0, R5 ; za 1 (u dvostrukoj preciznosti)

JR PETLJA

VAN POP R3

POP R2

POP R1

POP R0

RET

ORG 200

DW 77778888, 55556666 ; prvi broj

DW 33334444, 11112222 ; drugi broj

DW 0, 0 ; mjesto za rezultat

**4. (8,5 bodova)** FRISC prima 32-bitne NBC podatke koje mu šalje prekidna jedinica PVJ1 (spojena na INT). Primljeni podatci spremaju se kao bajtovi u memorijski blok na adresi 1000<sub>16</sub>. Ako primljeni podatak ne stane u 8-bita, treba prekinuti sa spremanjem i nakon toga zaustaviti glavni program. Ako se primi ništica, ona se ne sprema u memoriju, a primljene ništice treba prebrajati. Zanimarite mogućnost prepunjenja bloka podataka.

Glavni program cijelo vrijeme dojavljuje uvjetnoj jedinici UVJ broj ništica primljenih sa PVJ1.

Na NMI spojena je i PVJ3. Podatak koji se pročita iz PVJ3 treba poslati bezuvjetnoj jedinici BVJ.

Adrese vanjskih jedinica odaberite sami.

```
BVJ    EQU    0FFFF0000

UVJ_D  EQU    0FFFF0050
UVJ_S  EQU    0FFFF0054

PVJ1_D EQU    0FFFF1000
PVJ1_S EQU    0FFFF1004
PVJ1_E EQU    0FFFF1008

PVJ2_D EQU    0FFFF2000
PVJ2_S EQU    0FFFF2004
PVJ2_E EQU    0FFFF2008

PVJ3_D EQU    0FFFF3000
PVJ3_S EQU    0FFFF3004
PVJ3_E EQU    0FFFF3008

        ORG    0

        MOVE    10000, SP      ; inicijalizacija stoga
        JP      GLAVNI

        ORG     8
        DW      100           ; prekidni vektor

        ORG     0C           ; prekidni potprogram za NMI
        PUSH    R0
        STORE   R0, (PVJ3_S) ; dojavu prihvata prekida
        LOAD    R0, (PVJ3_D) ; pročitaj podatak sa PVJ3 i...
        STORE   R0, (BVJ)    ; ... pošalji ga bezuvjetnoj BVJ
        STORE   R0, (PVJ3_E) ; dojavu kraj posluživanja PVJ3
        POP     R0
        RETN

GLAVNI  MOVE    %B 10000, SR   ; dozvoli prekid INT

CEKAJ   LOAD    R0, (UVJ_S)    ; čekaj spremnost uvjetne UVJ
        OR      R0, R0, R0
        JR_Z    CEKAJ

        LOAD    R0, (NISTICE) ; pošalji uvjetnoj broj zanemarenih podataka
        STORE   R0, (UVJ_D)
        STORE   R0, (UVJ_S)    ; briši status uvjetnoj jedinici

        LOAD    R0, (STANI)    ; provjeri treba li zaustaviti glavni program
        OR      R0, R0, R0
        JR_Z    CEKAJ

KRAJ    HALT

; varijable razne

NISTICE    DW      0          ; brojač zanemarenih ništica
```

```

ADR_POD    DW      1000    ; adresa za spremanje u blok
STANI      DW      0       ; zastavica za zaustavljanje glavnog programa

ORG      100              ; prekidni potprogram za PVJ1
PUSH      R0
PUSH      R1
MOVE      SR, R0
PUSH      R0

STORE     R0, (PVJ1_S) ; dojaviti prihvati prekida PVJ1
LOAD      R0, (PVJ1_D) ; pročitaj podatak sa PVJ1

AND       R0, 0FFFFFF00, R1 ; provjeri 8-bitni opseg podatka
JR_NZ     STOP

CMP       R0, 0           ; provjeri ništicu
JR_EQ     NULA

SPREMI    LOAD      R1, (ADR_POD) ; učitaj adresu za pohranu u blok
STOREB    R0, (R1)           ; pohrani primljeni podatak u blok
ADD       R1, 1, R1          ; povećaj adresu za pohranu i spremi je natrag
STORE     R1, (ADR_POD)
JR        VAN

NULA      LOAD      R0, (NISTICE) ; povećati brojač zanemarenih ništica
ADD       R0, 1, R0
STORE     R0, (NISTICE)
JR        VAN

STOP      MOVE      1, R0
STORE     R0, (STANI)
JP        VAN2

VAN       STORE     R0, (PVJ1_E) ; dojaviti kraj obrade prekida PVJ1

VAN2      POP       R0
MOVE      R0, SR
POP       R1
POP       R0
RETI

```