1. međuispit iz Arhitekture računala 1 — 4. svibnja 2016.

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. 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 120 minuta.

1a. (1,5 bod) Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0101 - 1100. Nakon operacije će biti rezultat = _____ , a zastavice će biti prijenos = ____ , posudba = _____ , preljev = _____ , ništica = _____ , predznak = _____ . **MORA SE VIDJETI POSTUPAK RJEŠAVANJA, A NE SAMO REZULTATI !!!**

1b. (1 bod) U memoriji FRISC-a je na adresi 100_{16} zapisan podatak 11111111_2, a na adresi 101_{16} je zapisan 00000000_2.
Koji je to broj, ako je zapisan u 16-bitnom formatu dvojnog komplementa i redoslijedu big-endian: _____.
Koji je to broj, ako je zapisan u 16-bitnom formatu NBC i redoslijedu little-endian: _____.
REZULTATE PRIKAŽITE U DEKADSKOJ BAZI. MORA SE VIDJETI POSTUPAK RJEŠAVANJA!!!

1c. (2 boda) Napišite korake koje FRISC obavlja prilikom izvođenja naredbe SUB R0, 123, R1. 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:
_____
_____

1d. (2,5 boda) Odredite **trajanje u ciklusima** izvođenja sljedećeg **programa** (pretpostavite da je memorija **brza**):

		trajanje	*hazard*
TRI	EQU 3	_____	_____
	ORG 0	_____	_____
	MOVE TRI, R0	_____	_____
PETLJA	SUB R0, 1, R0	_____	_____
	STORE R0, (R0+1000)	_____	_____
	JR_NZ PETLJA	_____	_____
	HALT	_____	

Kraj **svake** naredbe napišite koliko puta se naredba izvodi po **koliko** ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c). Ako neka naredba izaziva hazard, napišite njegovo ime na drugoj crti. Izvođenje cijelog programa ukupno traje _____ ciklusa.

1e. (1 bod) Smjer FRISC-ovog priključka READ je _____, smjer priključka WRITE je _____. Smjer podatkovnih priključaka je _____, a adresnih je _____. Smjer priključka WAIT je _____.

1f. (3 boda) Prekidni priključci FRISC-a zovu se _____ i _____, od kojih _____ ima viši prioritet. Prioritetniji prekid naziva se _____ prekid, a manje prioritetan se naziva _____ prekid. Prihvaćanje manje prioritetnog prekida se može programski omogućiti ili onemogućiti pomoću zastavice _____ u registru _____. Prekidni potprogram za prioritetniji prekid nalazi se na adresi _____, a za manje prioritetan je na adresi _____. Naredbe RETI i RETN rade slično naredbi RET. Obje naredbe uzimaju povratnu adresu _____ (odakle) i stavljaju je u _____ (gdje). Osim toga, RETI dodatno upisuje vrijednost _____ u _____, a RETN dodatno upisuje vrijednost _____ u _____.

RJEŠENJA

1a. (1,5 bod) Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 0101 - 1100. Nakon operacije će biti rezultat = __1001__ , a zastavice će biti prijenos = __0_ , posudba = __1__ , preljev = __1__ , ništica = __0__ , predznak = __1___ . **MORA SE VIDJETI POSTUPAK RJEŠAVANJA, A NE SAMO REZULTATI !!!**

1b. (1 bod) U memoriji FRISC-a je na adresi 100_{16} zapisan podatak 11111111_2, a na adresi 101_{16} je zapisan 00000000_2.
Koji je to broj ako je zapisan u 16-bitnom formatu dvojnog komplementa i redoslijedu big-endian: ____-256___.
Koji je to broj ako je zapisan u 16-bitnom formatu NBC i redoslijedu little-endian: ____255____.
REZULTATE PRIKAŽITE U DEKADSKOJ BAZI. MORA SE VIDJETI POSTUPAK RJEŠAVANJA!!!

1c. (2 boda) Napišite korake koje FRISC obavlja prilikom izvođenja naredbe SUB R0, 123, R1. Ne treba popuniti sve crte.

Razina dohvata:
Prva polovina periode CLOCK-a:
__PC -> AR_____
Druga polovina periode CLOCK-a:
__(AR) -> IR_____
__dekodiranje_____
__operandi -> ALU__(ili R0, ext 123 -> ALU)_____
__izbor i pokretanje ALU operacije (oduzimanja)_____
___PC+4 -> PC_____
_____

Razina izvođenja:
Prva polovina periode CLOCK-a:
___ALU završava operaciju (oduzimanje), rezultat ->R1___
___stanje zastavica -> SR_____
_____
_____
Druga polovina periode CLOCK-a:
_____
_____

1d. (2,5 boda) Odredite **trajanje u ciklusima** izvođenja sljedećeg **programa** (pretpostavite da je memorija **brza**):

		trajanje	*hazard*	
TRI	EQU 3	_____0_____	_____	
	ORG 0	_____0_____	_____	
	MOVE TRI, R0	_____1 x 1_____	_____	
PETLJA	SUB R0, 1, R0	_____3 x 1_____	_____	
	STORE R0, (R0+1000)	_____3 x 2_____	_____strukturni_____	
	JR_NZ PETLJA	_____2 x 2 + 1 x 1_____	_____upravljački_____	(ili kontrolni)
	HALT	_____1 x 2_____		

Kraj **svake** naredbe napišite koliko puta se naredba izvodi po **koliko** ciklusa (npr. 6 x 2c ili 1 x 2c + 1 x 1c). Ako neka naredba izaziva hazard, napišite njegovo ime na drugoj crti. Izvođenje cijelog programa ukupno traje __17__ ciklusa.
umjesto 1x2 na naredbi HALT, može se računati punjenje na naredbi MOVE (1punjenje + 1x1)

1e. (1 bod) Smjer FRISC-ovog priključka READ je ___izlazni____, smjer priključka WRITE je ____izlazni____. Smjer podatkovnih priključaka je _____dvosmjerni__, a adresnih je _____izlazni___. Smjer priključka WAIT je __ulazni___.

1f. (3 boda) Prekidni priključci FRISC-a zovu se _INT_(ili int[0])_ i _NMI_(ili int[1])_, od kojih _NMI_ ima viši prioritet. Prioritetniji prekid naziva se __nemaskirajući_ prekid, a manje prioritetan se naziva ___maskirajući __ prekid. Prihvaćanje manje prioritetnog prekida se može programski omogućiti ili onemogućiti pomoću zastavice __GIE_ u registru _SR_. Prekidni potprogram za prioritetniji prekid nalazi se na adresi __C (ili 12)____, a za manje prioritetan je na adresi ____koja je zapisana (u vektoru) na lokaciji 8_____. Naredbe RETI i RETN rade slično naredbi RET. Obje naredbe uzimaju povratnu adresu __sa (vrha) stoga_____ (odakle) i stavljaju je u _PC_ (gdje). Osim toga, RETI dodatno upisuje vrijednost _1_ u _GIE_, a RETN dodatno upisuje vrijednost _1_ u _(internu zastavicu) IIF__.

2. (6 bodova) Napisati potprogram PREBROJI koji prima dva parametra: prvi je podatak, a drugi je širina podatka (širina je u opsegu od 1 do 32_{10}). Podatak se prima pomoću stoga, a širina se prima pomoću registra R0. Potprogram treba prebrojiti jedinice u onoliko nižih bitova podatka koliko je zadano širinom. Rezultat (broj jedinica) se vraća registrom R0.

Napisati potprogram PARITET, koji pomoću stoga prima 15-bitni podatak (viših 17 bita sadrže ništice). Potprogram treba šesnaesti bit podatka postaviti tako da rezultantni 16-bitni podatak ima parni paritet. Rezultantni 16-bitni podatak se vraća registrom R1. Jedinice u podatku potrebno je prebrojiti pomoću potprograma PREBROJI.

U memoriji se nalazi blok sa 300_{16} 16-bitnih podataka (koji imaju ništicu na najvišem bitu), a blok počinje na adresi 5000_{16}. Glavni program treba korištenjem potprograma PARITET promijeniti sve 16-bitne podatke iz bloka tako da im paritet bude paran.

Svi potprogrami moraju čuvati stanja registara.

```
       ORG 0

GLAVNI MOVE   10000, SP          ; inicijaliziraj stog, tj. SP
       MOVE   5000, R5           ; R5 adresira podatke u bloku
       MOVE   300, R3            ; brojač za petlju

PETLJA LOADH  R0, (R5)           ; dohvati originalni podatak
       PUSH   R0                 ; stavi ga kao parametar na stog
       CALL   PARITET            ; izračunaj podatak s parnim paritetom
       ADD    SP, 4, SP          ; ukloni parametar sa stoga

       STOREH R1, (R5)           ; spremi podatak s parnim paritetom preko originala
       ADD    R5, 2, R5          ; pomak adrese

       SUB    R3, 1, R3          ; smanjivanje brojača i ponavljanje petlje
       JR_NZ  PETLJA

       HALT

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

PARITET PUSH  R0                 ; spremi kontekst (R0 obavezno spremiti jer se poziva...
                                 ; ...PREBROJI koji ga mijenja (vraća rezultat u R0)

       LOAD   R1, (SP+8)         ; dohvati parametar sa stoga

       MOVE   %D 15, R0          ; širina podatka je jedan parametar
       PUSH   R1                 ; sam podatak je drugi parametar

       CALL   PREBROJI           ; prebroji jedinice, rezultat se vraća preko R0
       ADD    SP, 4, SP          ; ukloni parametar sa stoga

       AND    R0, 1, R0          ; ispitaj parnost rezultata (tj. ispitaj najniži bit)
       JR_Z   PARAN_PARIT

NEPARAN_PARIT
       OR     R1, 8000, R1       ; ako je paritet neparan, postavi šesnaesti bit

PARAN_PARIT                      ; ako je paritet već paran, ne treba raditi ništa
                                 ; nije greška ako se napravi AND R1, 7FFF, R1

       POP R0                    ; obnovi kontekst i vrati se
       RET
```

```
---------------------------------

PREBROJI PUSH R1                ; spremi kontekst
       PUSH   R2

       LOAD   R1, (SP+0C)       ; dohvati podatak u R1 (širina je prenesena sa R0)
       MOVE   0, R2             ; brojač jedinica R2 inicijalno postavi na 0

BROJI  ROTR   R1, 1, R1         ; ispitaj najniži bit i pomakni podatak
       JR_NC  DALJE

JEDAN  ADD    R2, 1, R2         ; ako je bit u jedinici, povećaj brojač R2

DALJE  SUB    R0, 1, R0         ; smanji brojač petlje i ponovi petlju
       JR_NZ  BROJI

       MOVE   R2, R0            ; stavi rezultat iz R2 u "povratni registar R0"

       POP    R2                ; obnovi kontekst i vrati se
       POP    R1
       RET

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

       ORG    5000
       DH     2, 74, 2, ....    ; blok sa 300 podataka
```

3. (8 bodova) Na FRISC su spojene dvije uvjetne vanjske jedinice U1 i U2, bezuvjetna jedinica BJ i prekidna jedinica PJ. Adrese im odaberite sami.

FRISC prenosi podatke sa U1 na U2. Sa U1 primaju se podatci koji u nižih 16-bita sadrže broj u formatu 2'k, dok su viših 16 bita u ništicama. Na U2 treba slati 32-bitne podatke u formatu s bitom za predznak. Kad god se na U2 šalje pozitivan podatak, na bezuvjetnu jedinicu BJ treba poslati podatak 0. Kada se na U2 šalje negativan podatak, onda na BJ treba poslati podatak 1.

Prenošenje sa U1 na U2 odvija se sve dok se od U1 ne primi podatak 8000_{16} - tada treba zaustaviti prekidnu jedinicu i procesor. Prilikom prenošenja podataka prebraja se koliko podataka je preneseno.

Prekidna jedinica PJ spojena je na maskirajući prekidni priključak INT. Kada se primi prekid, treba prekidnoj jedinici poslati broj do tada prenesenih podataka od U1 na U2. Brojač dodatno treba ispitati, i ako je postao strogo veći od 100_{16}, treba ga vratiti na ništicu.

```
BJ            EQU    0FFFF0000

U1_D          EQU    0FFFF1000
U1_BS         EQU    0FFFF1004

U2_D          EQU    0FFFF2000
U2_BS         EQU    0FFFF2004

PJ_D          EQU    0FFFF3000
PJ_BS         EQU    0FFFF3004
PJ_IEND       EQU    0FFFF3008
PJ_STOP       EQU    0FFFF300C

       ORG    0                  ; ORG 0 se može ispustiti
START  MOVE   10000, SP          ; početak izvođenja
       JR     GLAVNI

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

       ORG    08                 ; prekidni vektor na adresi 08
       DW     200

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

GLAVNI MOVE   %B 10000, SR       ; omogući prekid INT

CEK1   LOAD   R0, (U1_BS)        ; ispitaj i čekaj spremnost U1
       CMP    R0, 0
       JR_EQ  CEK1

       LOAD   R1, (U1_D)         ; primi podatak sa U1 (može i LOADH)
       STORE  R0, (U1_BS)        ; obriši status od U1

       CMP    R1, 8000           ; ispitaj oznaku kraja
       JR_EQ  KRAJ

       AND    R1, 8000, R0       ; ispitaj predznak 16-bitnog 2'k broja
       JR_Z   CEK2               ; ako je pozitivan, odmah se šalje

       ; pretvorba 16-bitni 2'k  --->  32-bitni bit za predznak
NEGAT  SHL    R1, %D 16, R1      ; predznačno proširi na 32-bita
       ASHR   R1, %D 16, R1
       XOR    R1, -1, R1         ; napravi dvojni komplement
       ADD    R1, 1, R1

       ; gornje 4 naredbe mogu i kraće (bez predznačnog proširenja):
       ; XOR R1, 0FFFF, R1
       ; ADD R1, 1, R1

       LOAD   R0, (PREDZNAK)     ; postavi bit predznaka
       OR, R1, R0, R1
```

```
       ; Predznak se također može postaviti na razne načine, npr:
       ; ROTL R1, 1, R1
       ; OR R1, 1, R1 (ili ADD ili XOR umjesto OR)
       ; ROTR R1, 1, R1

CEK2   LOAD   R0, (U2_BS)         ; ispitaj i čekaj spremnost U2
       CMP    R0, 0
       JR_EQ  CEK2

       STORE  R1, (U2_D)          ; šalji podatak na U2 i obriši joj status
       STORE  R0, (U2_BS)

       ; provjeri predznak rezultata i šalji na bezuvjetnu jedinicu BJ

       AND    R0, R0, R0          ; i ova provjera sa slanjem može kraće:
       JR_P   POZIT               ; ROTL R0, 1, R0 - predznak na najniže mjesto
                                  ; AND  R0, 1, R0 - brisanje ostalih bitova
NEG    MOVE   1, R1               ; STORE R0, (BJ)
       JP     SALJI

POZ    MOVE   0, R1
       JP     SALJI

SALJI  STORE R1, (BJ)             ; slanje 0 ili 1 na bezuvjetnu BJ

       LOAD   R1, (BROJAC)        ; povećaj brojač
       ADD    R1, 1, R1           ; BROJAČ MORA BITI U MEMORIJI, A NE U REGISTRU
       STORE  R1, (BROJAC)

       JR     CEK1         ; Natrag na posluživanje prve jedinice U1.
                           ; Bitan je redoslijed posluživanja: prvo čekanje na U1 pa onda
                           ; čekanje na U2, i na kraju povratak na čekanje U1.


       ; kraj u slučaju primitka podatka 8000
KRAJ   MOVE   0, R0
       STORE  R0, (PJ_STOP)       ; zaustavi prekidnu jedinicu PJ i procesor
       HALT

PREDZNAK      DW     80000000     ; maska za postavljanje predznaka (tj. najvišeg bita)
BROJAC        DW     0            ; brojač poslanih podataka

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

       ORG    200                 ; PREKIDNI POTPROGRAM

       PUSH   R0                  ; pohrani kontekst
       MOVE   SR, R0
       PUSH   R0

       STORE  R0, (PJ_BS)         ; dojava prihvata prekida

       LOAD   R0, (BROJAC)        ; čitanje brojača...
       STORE  R0, (PJ_D)          ; ...i slanje na PJ

       CMP    R0, 100             ; provjera je li brojač veći od 100
       JR_ULE NIJE                ; ako nije (manji je ili jednak od 100)

VECI   MOVE   0, R0               ; obriši brojač
       STORE  R0, (BROJAC)

NIJE   STORE  R0, (PJ_IEND)       ; dojavi kraj posluživanja

       POP    R0                  ; obnova konteksta
       MOVE   R0, SR
       POP    R0
       RETI                       ; povratak
```

4. (5 bodova) Na FRISC je spojena bezuvjetna jedinca BJ, te prekidna jedinica PJ na priključak NMI. Adrese im odaberite sami.

PJ šalje samo podatke 1, 2 i -1. Kad god PJ izazove prekid, treba s nje pročitati podatak i spremiti ga u memoriju na adresu 1000_{16}. Podatak pročitan sa PJ treba poslati na BJ samo ako je jednak 1 ili 2.

Glavni program za to vrijeme beskonačno dodaje sadržaj lokacije 1000_{16} lokaciji VAR. Zanemarite prekoračenje opsega lokacije VAR. Početne vrijednosti lokacije VAR i 1000_{16} trebaju biti 1. Kada glavni program prepozna da je na lokaciji 1000_{16} podatak -1, treba zaustaviti rad PJ i procesora.

```
BJ          EQU    0FFFF0000

PJ_D        EQU    0FFFF1000
PJ_BS       EQU    0FFFF1004
PJ_IEND     EQU    0FFFF1008
PJ_STOP     EQU    0FFFF100C

       ORG    0                  ; ORG 0 se može ispustiti
START  MOVE   10000, SP
       JR     GLAVNI

---------------------------------
       ORG    0C                 ; prekidni potprogram za NMI na adresi 0C

       PUSH   R0                 ; pohrani kontekst
       MOVE   SR, R0
       PUSH   R0

       STORE  R0, (PJ_BS)        ; dojava prihvata prekida

       LOAD   R0, (PJ_D)         ; primanje podatka od Prekidne PJ...
       STORE  R0, (1000)         ; ...i spremanje na 1000

       AND    R0, R0, R0         ; ispitivanje predznaka podatka
       JR_N   VAN

       STORE  R0, (BJ)           ; ako nije negativan, pošalji ga bezuvjetnoj BJ

VAN    STORE  R0, (PJ_IEND)      ; dojavi kraj posluživanja

       POP    R0                 ; obnova konteksta
       MOVE   R0, SR
       POP    R0
       RETN                      ; povratak

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

GLAVNI LOAD   R0, (VAR)          ; učitaj podatak sa lokacije VAR

       LOAD   R1, (1000)         ; učitaj podatak sa lokacije 1000
       CMP    R1, -1             ; provjeri je li kraj
       JR_EQ  KRAJ

       ADD    R0, R1, R0         ; uvećaj (VAR) za (1000)...
       STORE  R0, (VAR)          ; ...i spremi natrag na lokaciju VAR
                                 ; NE držati VAR u registru bez spremanja u memoriju!!!

       JR     GLAVNI             ; ponavljaj beskonačno

KRAJ   MOVE   0, R0
       STORE  R0, (PJ_STOP)      ; zaustavi prekidnu jedinicu
       HALT                      ; zaustavi i procesor

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

VAR    DW     1                  ; lokacija VAR za uvećavanje

       ORG    1000
       DW     1                  ; lokacija koja se pridodaje lokaciji VAR
```