1. međuispit iz Arhitekture računala 1 — 23. travnja 2015.

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 120 minuta.

1a. (2 boda) U memoriju čije riječi su široke 8 bita podatci se upisuju u redoslijedu **big-endian**. Od adrese 100_{16} upišite redom sljedeće podatke (tablicu popunite u **heksadekadskoj** brojevnoj bazi):

31_{10} u 8-bitnom formatu s bitom za predznak,
-31_{10} u 8-bitnom formatu s bitom za predznak,
260_{10} u 32-bitom formatu 2'k,
260_{10} u 16-bitnom formatu NBC,
31_{10} u 8-bitnom formatu 2'k,
-0 (negativna nula) u 8-bitnom formatu s bitom za predznak.

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

100	
101	
102	
103	
104	
105	
106	
107	
108	
109	

1b. (1,5 bod) Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 01010 - 11001. Nakon operacije će biti rezultat = _____, a zastavice će biti prijenos = ____, posudba = _____ , preljev = _____ , ništica = _____ , predznak = _____ . **Potrebno je napisati i postupak rješenja, a ne samo rezultate.**

1c. (2 boda) Napišite korake koje FRISC obavlja prilikom izvođenja naredbe ADD R0, 300, 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. (1 boda) Koliko razina (engl. stage) ima FRISC-ova protočna struktura: ____ . Uz pretpostavku da je memorija brza, naredba CALL_C efektivno traje ___ ciklusa kada je zastavica C jednaka ništici. Kad je zastavica C jednaka jedinici, naredba CALL_C efektivno traje ___ ciklusa. Hazard u naredbi CALL naziva se _____ .

1e. (2 boda, +0,25 za točni i -0,25 za svaki netočni odgovor). Zaokružite točne odgovore u sljedećim tvrdnjama.

1. FRISC ima zajedničku memorijsku i UI sabirnicu (tzv. backplane) / memorijsku i UI sabirnicu spojene pomoću međusklopa.
2. Za adresiranje UI-jedinica FRISC koristi memorijsko UI preslikavanje / izdvojeno UI adresiranje.
3. Čitanje iz sporih memorija može se produljiti za jedan ili više ciklusa CLOCK-a pomoću FRISC-ovog priključka READ / WAIT. Smjer navedenog priključka je ulazni / izlazni.
4. FRISC pristupa memorijama i UI jedinicama na jednaki / različit način.
5. FRISC koristi sinkrone / asinkrone protokole za komunikaciju pomoću sabirnica.
6. Adresni priključci kod FRISC-a su izlazni / dvosmjerni.
7. Asinkrone sabirnice obično se koriste kao memorijske sabirnice / UI sabirnice.

1.f. (2 boda) Za povratak iz **maskirajućeg** prekidnog potprograma koristi se naredba _____ koja u zastavicu ___ upisuje vrijednost ___. Za povratak iz **nemaskirajućeg** prekidnog potprograma koristi se naredba _____ koja u zastavicu ___ upisuje vrijednost ___. Obje naredbe uzimaju povratnu adresu _____ (odakle) i stavljaju je u ____ (gdje).

1.g. (0,5 boda) Kada se u potprogram parametri prenose stogom, u potprogramu ih sa stoga čitamo korištenjem adresiranja koje se zove _____. Pri tome se koristi registar _____.

2. (7,5 bodova) Napisati potprogram MNOZ koji metodom uzastopnog pribrajanja množi dva 16-bitna NBC broja i daje 32-bitni rezultat. Rezultat mora biti ispravan i kada je neki od brojeva jednak nuli. Brojevi se prenose u potprogram kao parametri preko stoga, a rezultat se vraća preko R0.

U memoriji od adrese 1000_{16} nalazi se blok podataka koji treba obraditi korištenjem potprograma MNOZ na sljedeći način. Svaki 32-bitni podatak u bloku sastoji se od četiriju 8-bitnih NBC-brojeva. Ta četiri broja treba pomnožiti, a ukupni 32-bitni umnožak treba spremiti preko originalnog podatka. Kraj bloka zaključen je (terminiran) podatkom $FFFFFFFF_{16}$.

3. (11,5 bodova) FRISC prima podatke s uvjetne jedinice UVJ1. Podatci su 32-bitni i u zapisu 2'k. Primljeni pozitivni podatci se prenose na drugu uvjetnu jedinicu UVJ2. Negativni podatci se šalju na bezuvjetnu jedinicu BVJ. Pri tome se u memorijskim lokacijama posebno prebrajaju poslani pozitivni i posebno poslani negativni podaci. Kad se primi podatak 0, treba zaustaviti procesor.

Na priključak INT spojena je prekidna jedinica PVJ1. Kad ona izazove prekid, treba joj poslati broj 1 ako je od UVJ1 primljeno više pozitivnih podataka, odnosno -1 ako je primljeno više ili jednako negativnih. Također treba resetirati brojače pozitivnih i negativnih podataka.

Na priključak NMI spojena je prekidna jedinica PVJ2. Kad ona izazove prekid, treba joj poslati vrijednost brojača negativnih podataka.

1a. (2 boda) U memoriju čije riječi su široke 8 bita podatci se upisuju u redoslijedu **big-endian**. Od adrese 100_{16} upišite redom sljedeće podatke (tablicu popunite u **heksadekadskoj** brojevnoj bazi):

31_{10} u 8-bitnom formatu s bitom za predznak,
-31_{10} u 8-bitnom formatu s bitom za predznak,
$+260_{10}$ u 32-bitom formatu 2'k.
260_{10} u 16-bitnom formatu NBC,
31_{10} u 8-bitnom formatu 2'k,
-0 (negativna nula) u 8-bitnom formatu s bitom za predznak.

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

100	1F
101	9F
102	00
103	00
104	01
105	04
106	01
107	04
108	1F
109	80

1b. (1,5 bod) Općenita 4-bitna (ne FRISC-ova) aritmetičko-logička jedinica oduzima binarne brojeve 01010 - 11001. Nakon operacije će biti rezultat = _0001/10001_, a zastavice će biti prijenos = _1/0_, posudba = _0/1_ , preljev = _0/1_ , ništica = __0/0_ , predznak = __0/1__ . **Potrebno je napisati postupak rješenja, a ne samo rezultate.**

1c. (2 boda) Napišite korake koje FRISC obavlja prilikom izvođenja naredbe ADD R0, 300, R1. 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 R0 -> ALU _____

___izbor i pokretanje ALU-operacije ____

___PC +4 -> PC_____

Razina izvođenja:

Prva polovina periode CLOCK-a:

___ALU -> R1 _____

___zastavice -> SR_____

Druga polovina periode CLOCK-a:

_____

_____

_____

_____

1d. (1 boda) Koliko razina (engl. stage) ima FRISC-ova protočna struktura: __2__ . Uz pretpostavku da je memorija brza, naredba CALL_C efektivno traje _1__ ciklusa kada je zastavica C jednaka ništici. Kad je zastavica C jednaka jedinici, naredba CALL_C efektivno traje _2__ ciklusa. Hazard u naredbi CALL naziva se _____UPRAVLJAČKI_____ .

1e. (2 boda). Zaokružite točne odgovore u sljedećim tvrdnjama.

1. FRISC koristi zajedničku memorijsku i UI sabirnicu (tzv. backplane) / memorijsku i UI sabirnicu spojene pomoću međusklopa.
2. Za adresiranje UI-jedinica FRISC koristi memorijsko UI preslikavanje / izdvojeno UI adresiranje.
3. Čitanje iz sporih memorija može se produljiti za jedan ili više ciklusa CLOCK-a pomoću FRISC-ovog priključka READ / WAIT. Smjer navedenog priključka je ulazni / izlazni.
4. FRISC pristupa memorijama i UI jedinicama na jednaki / različit način.
5. FRISC koristi sinkrone / asinkrone protokole za komunikaciju pomoću sabirnica.
6. Adresni priključci kod FRISC-a su izlazni / dvosmjerni.
7. Asinkrone sabirnice obično se koriste kao memorijske sabirnice / UI sabirnice.

1.f. (2 boda) Za povratak iz **maskirajućeg** prekidnog potprograma koristi se naredba RETI_ koja u zastavicu _GIE_ upisuje vrijednost _1_. Za povratak iz **nemaskirajućeg** prekidnog potprograma koristi se naredba _RETN_ koja u zastavicu _IIF_ upisuje vrijednost _1_. Obje naredbe uzimaju povratnu adresu _sa stoga_ (odakle) i stavljaju je u _PC_ (gdje).

1.g. (0,5 boda) Kada se u potprogram parametri prenose stogom, u potprogramu ih sa stoga čitamo korištenjem adresiranja koje se zove _registarsko indirektno s odmakom (ili ofsetom)_. Pri tome se koristi registar _SP (ili R7)_.

2. (7,5 bodova) Napisati potprogram MNOZ koji metodom uzastopnog pribrajanja množi dva 16-bitna NBC broja i daje 32-bitni rezultat. Rezultat mora biti ispravan i kada je neki od brojeva jednak nuli. Brojevi se prenose u potprogram kao parametri preko stoga, a rezultat se vraća preko R0.

U memoriji od adrese 1000_{16} nalazi se blok podataka koji treba obraditi korištenjem potprograma MNOZ na sljedeći način. Svaki 32-bitni podatak u bloku sastoji se od četiriju 8-bitnih NBC-brojeva. Ta četiri broja treba pomnožiti, a ukupni 32-bitni umnožak treba spremiti preko originalnog podatka. Kraj bloka zaključen je (terminiran) podatkom $FFFFFFFF_{16}$.

```
       ORG 0

GLAVNI MOVE   10000, SP          ; inicijaliziraj stog, tj. SP
       MOVE   1000, R6           ; R6 adresira podatke u bloku

PETLJA LOAD   R0, (R6)           ; učitaj podatak iz 1. bloka i
       CMP    R0, -1             ; provjeri je li zaključni podatak
       JR_EQ  KRAJ               ; ako jeste => kraj petlje

       LOADB  R1, (R6)           ; izdvoji prva dva bajta
       LOADB  R2, (R6+1)

; izdvajanje bajtova može na razne načine, na primjer ovako:
;      LOAD   R0, (R6)
;      AND    R0, 0FF, R1
;      AND    R0, 0FF00, R2
;      SHR    R2, 8, R2

       PUSH   R1                 ; prva dva bajta stavi na stog za množenje
       PUSH   R2
       CALL   MNOZ               ; pomnoži ih
       ADD    SP, 8, SP          ; ukloni parametre sa stoga
       MOVE   R0, R3             ; zapamti rezultat (bolje je odmah PUSH R0)

       LOADB  R1, (R6+2)         ; izdvoji druga dva bajta
       LOADB  R2, (R6+3)

       PUSH   R1                 ; druga dva bajta stavi na stog za množenje
       PUSH   R2
       CALL   MNOZ               ; pomnoži ih
       ADD    SP, 8, SP          ; ukloni parametre sa stoga

       PUSH   R0                 ; međurezultate stavi na stog za množenje
       PUSH   R3
       CALL   MNOZ               ; pomnoži ih
       ADD    SP, 8, SP          ; ukloni parametre sa stoga

       STORE  R0, (R6)           ; 32-bitni ukupni umnožak prepiši preko originalnog
                                 ; podatka u bloku

       ADD    R6, 4, R6          ; pomakni adresu na sljedeća 4 bajta u bloku
       JR     LOOP               ; natrag na početak petlje

KRAJ   HALT                      ; kraj programa
```

nastavak rješenja je na drugoj strani

```
MNOZ   PUSH   R1                 ; spremi kontekst
       PUSH   R2

       LOAD   R1, (SP + %D 12)    ; dohvati parametre sa stoga
       LOAD   R2, (SP + %D 16)

       MOVE   0, R0               ; početna vrijednost rezultata

LOOPM  CMP    R2, 0               ; provjeri je li kraj petlje
       JR_EQ  VAN

       ADD    R0, R1, R0          ; dodaj jedan parametar na rezultat
       SUB    R2, 1, R2           ; smanji drugi parametar i ponavljaj
       JR     LOOPM

       POP    R2                  ; obnova konteksta
       POP    R1

       RET                        ; povratak
```

3. (11,5 bodova) FRISC prima podatke sa uvjetne jedinice UVJ1. Podatci su 32-bitni i u zapisu 2'k. Primljeni pozitivni podatci se prenose na drugu uvjetnu jedinicu UVJ2. Negativni podatci se šalju na bezuvjetnu jedinicu BVJ. Pri tome se u memorijskim lokacijama posebno prebrajaju poslani pozitivni i posebno poslani negativni podaci. Kad se primi podatak 0, treba zaustaviti procesor.

Na priključak INT spojena je prekidna jedinica PVJ1. Kad ona izazove prekid, treba joj poslati broj 1 ako je od UVJ1 primljeno više pozitivnih podataka, odnosno -1 ako je primljeno više ili jednako negativnih. Također treba resetirati brojače pozitivnih i negativnih podataka.

Na priključak NMI spojena je prekidna jedinica PVJ2. Kad ona izazove prekid, treba joj poslati vrijednost brojača negativnih podataka.

```
BVJ    EQU    0FFFF0000    ; adrese svih vanjskih jedinica

UVJ1_D EQU    0FFFF0050
UVJ1_S EQU    0FFFF0054

UVJ2_D EQU    0FFFF0050
UVJ2_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

       ORG    0

       MOVE   10000, SP    ; inicijalizacija stoga
       JP     GLAVNI       ; prelazak u glavni


       ORG    8            ; adresa vektora
       DW     1000         ; prekidni vektor


       ORG    0C           ; adresa prekidnog potprograma za NMI

       PUSH   R0                  ; spremi kontekst

       STORE  R0, (PVJ2_S)        ; dojavi prihvat prekida

       LOAD   R0, (BROJAC_NEG)    ; pročitaj brojač negativnih
       STORE  R0, (PVJ2_D)        ; ... i pošalji ga na PVJ2

       STORE  R0, (PVJ2_E)        ; dojavi kraj posluživanja PVJ2

       POP R0                     ; obnovi kontekst
       RETN                       ; povratak iz nemaskirajućeg p.p.
```

nastavak rješenja je na drugoj strani

```
GLAVNI MOVE %B 10000, SR          ; dozvoli prekid INT

CEK1   LOAD   R0, (UVJ1_S)        ; čekaj spremnost uvjetne UVJ1
       CMP    R0, 0
       JR_EQ  CEK1

       LOAD   R3, (UVJ1_D)        ; pročitaj podatak iz UVJ1
       STORE  R0, (UVJ1_S)        ; briši status jedinici UVJ1

CHK    CMP    R3, 0               ; provjera primljenog podatka
       JR_EQ  KRAJ                ; ako je primljena nula => kraj programa
       JR_SGT POZIT               ; inače, ako je pozitivan => idi na POZIT


       ; inače, podatak je negativan, treba ga poslati na bezuvjetnu BVJ
NEGAT  STORE  R3, (BVJ)           ; šalji ga bezuvjetno na BVJ

       LOAD   R3, (BROJAC_NEG)    ; povećaj brojač negativnih podataka
       ADD    R3, 1, R3
       STORE  R3, (BROJAC_NEG)

       JR     CEK1                ; natrag na primanje podataka od UVJ1


       ; podatak je pozitivan, treba ga poslati na uvjetnu UVJ2
POZIT  LOAD   R0, (UVJ2_S)        ; čekaj spremnost uvjetne UVJ2
       CMP    R0, 0
       JR_EQ  POZIT

       STORE  R3, (UVJ2_D)        ; pošalji podatak jedinici UVJ2
       STORE  R0, (UVJ2_S)        ; briši status jedinici UVJ2

       LOAD   R3, (BROJAC_POZ)    ; povećaj brojač pozitivnih podataka
       ADD    R3, 1, R3
       STORE  R3, (BROJAC_POZ)
       JR     CEK1                ; natrag na primanje podataka od UVJ1

       ; ovdje se dođe kad se od UVJ1 primi nula, tada treba zaustaviti program
KRAJ   HALT                       ; zaustavi procesor




; lokacije koje služe kao brojači pozitivnih i negativnih podataka, inicijalno su 0

BROJAC_POZ    DW     0      ; brojač pozitivnih podataka
BROJAC_NEG    DW     0      ; brojač negativnih podataka
```

nastavak rješenja je na drugoj strani

```
       ORG    1000                ; adresa prekidnog potprograma za PVJ1

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

       STORE  R0, (PVJ1_S)        ; dojavi prihvata prekida jedinici PVJ1


       LOAD   R0, (BROJAC_POZ)    ; pročitaj brojače podataka u R0 i R1
       LOAD   R1, (BROJAC_NEG)

       CMP    R0, R1              ; usporedi brojače
       JR_UGT VISE_POZ

VISE_JEDNAKO_NEGAT                ; primljeno je vise ili jednako negativnih
       MOVE   -1, R0              ; treba poslati -1
       JR     SALJI

VISE_POZIT                        ; primljeno je vise pozitivnih
       MOVE   +1, R0              ; treba poslati +1

SALJI  STORE  R0, (PVJ1_D)        ; pošalji -1 ili +1 na PVJ1


RESET  MOVE   0, R0               ; resetiraj (obriši) brojače pozitivnih i negativnih
       STORE  R0, (BROJAC_POZ)
       STORE  R0, (BROJAC_NEG)

       STORE  R0, (PVJ1_E)        ; dojava kraja posluživanja jedinici PVJ1

IZLAZ  POP    R0                  ; obnova konteksta
       MOVE   R0, SR
       POP    R1
       POP    R0
       RETI                ; povratak iz maskirajućeg p.p.
```