1.a. Podatak 0001112 u 6-bitnom NBC-u predstavlja broj _**7**_, a u 6-bitnom formatu 2'k predstavlja broj _**7**_. Podatak 11002 u 4-bitnom NBC-u predstavlja broj _**12**_, a u 4-bitnom formatu 2'k predstavlja _**-4**_.
1.b. Sabirnice se prema **namjeni** dijele na: _**adresnu**_, _**podatkovnu**_ i _**upravljačku (kontrolnu)**_. Prema **načinu komunikacije** sabirnice se dijele na _**sinkronu**_ i _ **asinkronu**_.
1.c. Koji **dio vanjske jedinice** postoji unutar uvjetnih i prekidnih, a ne postoji unutar bezuvjetnih jedinica: _**bistabil stanja (status bistabil)**_. **Priključci** koji postoje kod uvjetnih i prekidnih vanjskih jedinica (a ne postoje kod bezuvjetnih) nazivaju se _**priključci za sinkronizaciju (ili handshaking, ili rukovanje, ili READY i STROBE)**_. Ovi priključci povezuju _**vanjsku jedinicu**_ i _**vanjski proces (ili vanjski uređaj ili uređaj)** _.
2.a. FRISC Prilikom prihvadanja maksirajudeg prekida FRISC mijenja zastavicu _**GIE**_ u registru _**SR**_ čime se (postiže što) _**zabranjuje prihvadanje maskirajudih prekida**_. Za razliku od obične naredbe RET, naredba RETI **dodatno** (radi što) _**obnavlja zastavicu GIE** _, a naredba RETN **dodatno** (radi što) _**obnavlja (internu) zastavicu IIF**_. Sve tri naredbe RET, RETI i RETN u registar PC stavljaju _**povratnu adresu (sa stoga)**_.
2.b. FRISC Napišite **smjerove** sljededih priključaka procesora FRISC: ADR je _**izlazni**_, DATA je _**dvosmjerni**_, READ je _**izlazni**_, WRITE je _ **izlazni** _, WAIT je_ **ulazni** _, BREQ je _ **ulazni** _, BACK je _ **izlazni** _, SIZE je _ **izlazni** _. Čemu služi priključak WAIT? _**pomoću njega memorija (ili VJ) dojavljuje da je spora (ili traži umetanje ciklusa čekanja, ili traži od FRISC-a da pričeka itd.)**_
2.c. FRISC Koja su 4 načina rada sklopa FRISC-PIO: **ulazni, izlazni, postavljanje bitova, ispitivanje bitova**
2.d. FRISC Kad u sklopu FRISC-CT vrijednost u brojilu postane nula, događa se sljedede: _**brojilo se ponovno postavlja na početnu vrijednost (ili vrijednost limit registra LR kopira se u brojilo), CT postaje spreman, CT može postaviti prekid, generira se impuls na izlaznom priključku ZC_.**
3.a. ARM Za procesor ARM7 napišite **trajanjakoraka izvođenja** (u ciklusima) sljededih naredaba:

naredbe za obradu podataka	__**1**__	naredba LDM R13,{R1,R2,R14}	__**5**__
naredba LDR	__**3**__	naredba B s istinitim uvjetom skoka	__**3**__
naredba STR	__**2**__	naredba BL s lažnim uvjetom skoka	__**1**__

3.b. ARM Procesor ARM sa **statičkim predviđanjem grananja** izvodi sljededi programski odsječak:
LABELA1 B LABELA2
LABELA2 B LABELA1
Zaokružite točan odgovor (a ili b):
Za **prvu** naredbu predvidjet će se da: a) će se grananje dogoditi **b) se grananje nede dogoditi**
Za **drugu** naredbu predvidjet će se da: **a) će se grananje dogoditi** b) se grananje nede dogoditi
3.c. ARM ARM ima dva ulazna priključka za prekide: _**IRQ**_ i _**FIQ**_. Za obične prekide adresa prekidnog potprograma je _**1816**_, a za brze prekide adresa je _**1C16**_. Povratak iz prekidnog potprograma izvodi se naredbom SUBS PC,LR,#4, koja obnavlja sadržaje registra (ili više njih): _**PC (ili R15) i CPSR**_.
3.d. ARM Kada se pojavi impuls na priključku CLK1HZ (ARM-ovog sklopa RTC), što se dogodi s brojilom? _**brojilo se poveda za jedan**_. Kada vrijednost u brojilu postane jednaka (*čemu*) _ **vrijednosti u registru usporedbe (MR)** _, tada se u RTC-u automatski događa sljedede: _**RTC postaje spreman i RTC može postaviti zahtjev za prekid** _.

1 a) Sljedećim odsječkom želi se postići **kašnjenje od 8 sekundi** (uz pretpostavku da FRISC radi na 10 MHz). Trajanje prve naredbe je zanemarivo u odnosu na trajanje petlje. Napišite **trajanja** pojedinih naredaba (u ciklusima) i izračunajte koja **vrijednost** mora biti zapisana na lokaciji KONST.

		trajanje u ciklusima
	LOAD R0,(KONST)	__**2**__
PETLJA	LOAD R1,(BROJAC)	__**2**__
	ADD R1,1,R1	__**1**__
	STORE R1,(BROJAC)	__**2**__
	CMP R1,R0	__**1**__
	JR_NE PETLJA	__**2**__
	KONST DW __**%D 10 000 000**__	
	BROJAC DW 0	

1 b) U memoriji FRISC-a zapisan je 16-bitni broj u formatu big-endian: na adresi 10016 zapisano je 111111102, a na adresi 10116 zapisano je 111111002.Koji je to broj ako ga promatramo kao 16-bitni zapis dvojnog komplementa _**260**_.
1 c) Broj razina protočne strukture FRISC-a je _**2**_ . Naredba se dekodira u razini _**1**_**-(dohvata - može i samo broj)**_. Dvije vrste hazarda kod FRISC-a su: _**strukturni**_ i_**upravljački**_. Postoji još i_**podatkovni**_ hazard, ali do njega ne dolazi kod FRISC-a.
1 d) Kod FRISC-a postoje dvije vrste prekida:
1) _**maskirajući**_ koji dolaze preko priključaka _**INT0_INT1_i_INT2**_ i
2) _**nemaskirajući**_ koji dolaze preko _**INT3**_. Zastavica GIE nalazi se u registru _**SR**_ i ako je u ništici, onda su _**maskirajući**_ prekidi _**zabranjeni/onemogućeni/maskirani**_.

1 e) Za 5-bitne brojeve izvodi se aritmetička operacija. Odredite rezultat operacije i vrijednost prijenosa, preljeva, posudbe ništice i predznaka (**općenito**, NE za FRISC). **Potrebno je napisati postupak rješenja**.

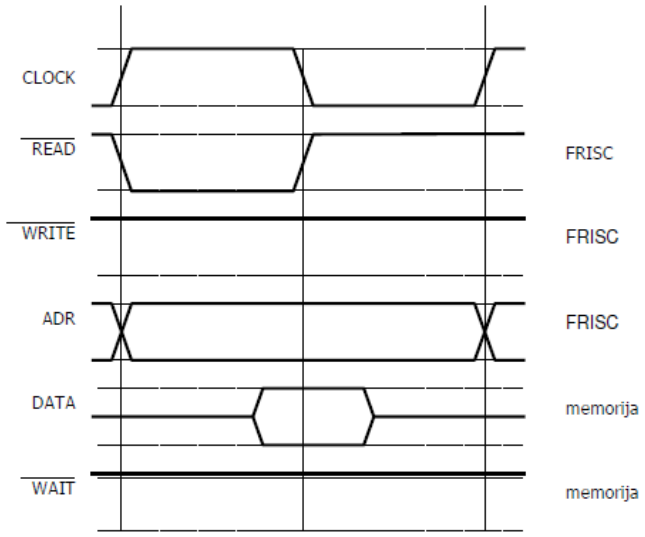
	prijenos	posudba	preljev	ništica	predznak
10110+10011 = __01001__	__1__	__0__	__1__	__0__	__0__
00110-11010 = __01100__	__0__	__1__	__0__	__0__	__0__

1 f) Na prazne crte upišite korake koje **FRISC** obavlja prilikom izvođenja naredbe **CMP R1,35**. Ne moraju se popuniti sve crte.

Razina dohvata:
Rastući brid CLOCK-a:
____PC →AR____
Padajući brid CLOCK-a:
____PC+4→ PC____
____(AR) →IR, dekodiranje____
____R1 i ext 35 _ ALU____

Razina izvođenja:
Rastući brid CLOCK-a:
____ALU: izvodi oduzimanje____
Padajući brid CLOCK-a:
____postavljanje zastavica u SR-u____

1 g) (3 boda) Nacrtajte signale na sabirnicama prilikom čitanja iz brze memorije kod FRISC-a. Napišite (na crte s desna) tko upravlja dotičnom sabirnicom.



1.a. FRISC Koji će biti sadržaj registra R0, nakon izvođenja naredbe LOAD na FRISCu za programski odsječak s desne strane: __**00560034**__

```
`ORG 130
DH 34, 56
DW 8558
...
LOAD R0, (130)
```

1.b. FRISC 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 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 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 **_000011112_** na adresu **_ FFFF4000_**i zatim podatak **_000001112_** na adresu **_ FFFF4000_**.

2.a. ARM 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 Procesor ARM izvodi naredbu LDMFD R13!, {R2,R1}. Ako je sadržaj registra R13 prije izvođenja naredbe bio 10010, sadržaj registra R13 nakon naredbe će biti _**10810**_ , a registar R1 će se napuniti podatkom s memorijske lokacije_**10010**_.

2.c. ARM 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 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 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 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 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)**

1 a) 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	**00**
F1	**00**
F2	**00**
F3	**00**
F4	**14**
F5	**00**
F6	**00**
F7	**00**
➔ F8	**10**
F9	**00**
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**

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) 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) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: ADR , DATA , READ , _WRITE_ , _ WAIT_ i _SIZE_.
1 d) Za procesor FRISC, kod naredbe LOAD dolazi do pojave _strukturnog_ hazarda, a kod naredbe PUSH dolazi do pojave _strukturnog_ hazarda.
1 e) 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) Odredite trajanje izvođenja sljedećeg programskog odsječka (pretpostavite da je memorija brza):

DVA	`EQU 2	Trajanje_**0**_	Izvodi se_**0**_ puta
	`ORG 0	Trajanje_**0**_	Izvodi se_**0**_ puta
	MOVE 4, R0	Trajanje_**1**_	Izvodi se_**1**_ puta
POC	SUB R0, DVA, R0	Trajanje_**1**_	Izvodi se_**2**_ puta
	JR_NE POC	Trajanje_**2**_	Izvodi se_**2**_ puta
	STORE R0, (1000)	Trajanje_**2**_	Izvodi se_**1**_ puta
	HALT	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.

1. Za procesor ARM napišite programski odsječak koji izvodi predznačno proširenje broja u registru R0 iz 8-bitnog zapisa 2'k, na 32 bita. Upotrijebite dvije naredbe MOV.
MOV R0, R0, LSL #18
MOV R0, R0, ASR #18
2. Za sljedede neposredne vrijednosti navedite je li ih mogude upisati kao dio naredbe procesora ARM.
0x02080000 _____ 0x0000FFFF _____
0xFFFF34FF _____ 0x00000101 _____
3. Kod procesora FRISC, uvjetno izvođenje naredbi mogude je za **_UPRAVLJAČKE_** naredbe.
Kod procesora ARM, uvjetno izvođenje naredbi mogude je za **_SKORO SVE_** naredbe.
4. Na prazne crte upišite korake koje FRISC obavlja prilikom izvođenja naredbe **JP_NV 500**:

Razina dohvata:
Rastudi brid CLOCK-a:
____PC −> AR____
Padajudi brid CLOCK-a:
____PC+4 −> PC____
____(AR) −> IR, dekodiranje____
____ispitivanje istinitosti uvjeta V=0____
____ako V=0: ext 500____
____onemogudi dohvat u sljededem ciklusu____

Razina izvođenja:
Rastudi brid CLOCK-a:
____PRAZNO____
Padajudi brid CLOCK-a:
____ako V=0: ext 500 −> PC____
____omogudi dohvat u sljededem ciklusu____

5. S obzirom na smještaj operanada postoje procesorske ahitekture: **_stogovna, akumulatorska, registar – memorija, registar-registar_.** Za RISC procesore, od ove 4 arhitekture uobičajeno se koristi **_registar-registar_**.
6. Procesor ARM izvodi sljededu instrukciju LDM R13, {R1} , pri čemu su početni sadržaji registra R13=10016 i R1=0102030416 . Sadržaj memorijskih lokacija prikazan je na slici. Odredite sadržaje registara R13 i R1 nakon izvođenja instrukcije.

Adresa	Podatak	Adresa	Podatak
103	04	0FE	09
102	05	0FD	0A
101	06	0FC	0B
100	07	0FB	0C
0FF	08	0FA	0D

R13=__**100**__
R1=__**0708090A**__

7. Dopunite stanje signala i sadržaje registara za sljedede korake kod prihvadanja INT3 na FRISC-u.
IACK ← _**1**_
IIF ← _**0**_
_**SP**_ ← PC
IACK ← _**0** _
PC ← _**12(dek)**_
1.a. 10112 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. 00112 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. 00112 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. 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. 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. 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. 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. 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. 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. 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	**0x19**
0x59	**0x29**
0x90	**0x00**

1.a. Broj -5 treba zapisati u 16-bitnom formatu 2'k u memoriju FRISC-a od adrese 10016. U bajtu na adresi 10016 pisat će _**FB** _, a u bajt na adresi 10116 pisat će _**FF** _ (prikažite bajtove u heksadekadskoj bazi).
1.b. 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. 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. 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. 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. 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. 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. 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. Nabrojite dvije naredbe za pristup registrima stanja procesora ARM: _**MRS**_, _**MSR**_.

2.e. Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak _**88776655**_, a u registru R1 podatak _**100**_ .

```
`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. Neka priručna memorija ima $\mathbf{32_{10}}$ 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	**0x2**
0x21	**0x1**
0x22	**0x2**
0x153	**0x13 (ili 19_{10})**

2.g. 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. 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. 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. Na stog podatke spremamo podatke naredbom STMFD. Ako ih želimo pročitati sa stoga u iste registre trebamo koristiti naredbu: _**LDMFD (ILI LDMIA)**_.

1. 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. 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. 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. 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. Je li broj 204_{16} moguće upisati kao neposrednu vrijednost kod aritmetičke naredbe procesora ARM? **Da** - Ne
6.Na stog podatke spremamo podatke naredbom STMFD. Ako ih želimo pročitati sa stoga u iste registre trebamo koristiti naredbu:_**LDMFD**, **LDMIA**_.
7. Nastavak S u naredbi SUBS PC, R14, #4 znači da treba registar _**SPSR**_upisati u registar _**CPSR**_.
8. 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. Za sklop FRISC-PIO vrijedi (zaokružite točne tvrdnje – **Da** ili **Ne**

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. 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. 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. Bistabil stanja postoji u (zaokružite točne tvrdnje – **Da** ili **Ne**
Bezuvjetne vanjske jedinice **NE**
UVJ, PVJ, FRISC-CT, FRISC-PIO, FRISC-DMA **DA**
13. Napišite primjer naredbe FRISC-a 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. 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. 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).

1. 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. 01011_2 je 5-bitni prikaz u formatu dvojnog komplementa. Prikažite taj broj u 6-bitnom formatu s bitom za predznak_**001011**_. 110112 je prikaz u 5-bitnom formatu dvojnog komplementa. Prikažite taj broj u 4-bitnom formatu jediničnog komplementa_**1010**_.
3. 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. 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

SP -> _**104**_

FB	itd
FC	**00**
FD	**00**
FE	**00**
FF	**00**
100	**78**
101	**56**
102	**34**
103	**12**
104	**CC**
105	**BB**
106	**AA**
107	**99**
108	**00**
109	itd

5. 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. 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. 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 (prekida):	maskirajućih	**nemaskirajućih**	svih
Sabirnica kod FRISC-a je:	**sinkrona**	asinkrona	niti jedno
Sabirnica kod FRISC-a je:	multipleksirana	**nemultipleksirana**	niti jedno
Koji od ovih priključaka su nužni za komunikaciju s memorijom:	**SIZE**	**READ**	**ADR**

8. Kada u sklopu FRISC-CT brojilo odbroji zadani broj ciklusa, događa se sljedeće (bilo je ponuđeno, pa sam napisao točna rješenja, nije bitan redoslijed događanja): _**u brojilo se automatski upiše vrijednost iz LR, brojilo automatski nastavlja s brojenjem, CT generira impuls na priključku ZC, CT može generirati zahtjev za prekid i CT postaje spreman ako je prethodno bio poslužen do kraja**_.

9. Priključci READY i STROBE nazivaju se priključcima za _**sinkonizaciju (rukovanje, handshaking)**_, a od FRISCovih vanjskih jedinica ima ih sklop _**PIO**_.

10. Procesor ARM povratnu adresu iz potprograma sprema u _**LR (R14)**_.

11. 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. Procesor ARM izvršava naredbu **RSB R0, R0, R0 LSL #4**. Navedena naredba ima funkciju _ **množenja registra R0 sa 15**_.

13. 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. U sustavu procesora ARM signali *HADDR* su dio sabirnice _**AHB**_ i širine su _**32**_ bita.

15. Nabrojite dvije naredbe za pristup registrima stanja procesora ARM: _**MRS**_, _**MSR**_.

16. Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak _**88776655**_, a u registru R1 podatak _**104**_.

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

17. Nakon uključenja procesor ARM izvodi programski odsječak s desne strane. Nakon izvođenja odsječka, u registru R0 se nalazi podatak _**11**_, a u registru R1 podatak _**104**_ .

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

1 a) 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) 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) Troprolazni asembleri, za razliku od dvoprolaznih asemblera, mogu prevoditi programe koji koriste _**makronaredbe**_.
1 d) 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) Osim CLOCK-a, FRISC kod pristupa memoriji koristi i priključke: _**ADR**_ , _**DATA**_ , _**READ**_ , _**WRITE**_ , **WAIT** i _**SIZE**_.
1 f) 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) DMA može obavljati prijenos na četiri načina. Njihovi nazivi su: _**krađa ciklusa**_, _**zaustavljanje procesora**_, _**blokovski prijenos**_ i _**multipleksirani prijenos**_.
1 h) 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) Za ARM-ove vanjske jedinice zaokružite točne odgovore.

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

1 j) 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) 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) 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) 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) 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**

sad 2006 pa meduispite