UPOREDBA FRISC-a I ARM-a	FRISC	ARM
VRSTA	ugradbeno računalo	ugradbeno računalo
RAZINE APSTRAKCIJE	razina arhitekture skupa naredaba i mikroarhitekture	
PRISTUP MEMORIJI	von Neumanov	
ARHITEKTURA SKUPA NAREDABA (S OBZIROM NA SMJEŠTAJ OPERANADA)	load-store arhitektura (reg-reg)	load-store arhitektura (reg-reg)
MEMORIJSKI PODSUSTAV	Adresna sabirnica A[20:0] - određuje mem. lokaciju kojoj se pristupa Podatkovna sabirnica D[31:0] - služi za prijenos podataka između procesora i memorije Upravljačka sabirnica (RD, WR, SIZE) - upravlja prijenosom podataka	
SKUP NAREDABA PROCESORA	RISC	RISC (ima i karakteristike CISC-a)
BROJ REGISTARA	8 (R0-R7, R7 tj SP služi za stog) i SR (registar stanja) ... zastavice Z, V, C i N se nalaze od 3-0 bita	16 (R0-R15, R13 tj LR služi za stog, R14 tj LR sadrži pov. adresu iz potprograma, a R15 ili PC sadrži adresu iduće naredbe u glavnom programu) i CPSR, SPSR (registri stanja) ... zastavice Z, V, C i N se nalaze od 31-28 bita
SKUP NAREDBI	8 vrsta naredbi, 3 osnovne: aritmetičko-logičke (ADD, SUB, AND, OR, XOR), memorijske (LOAD, STORE), upravljačke (JP, JP_uvjet) naredbe; i 5 proširenja: aritmetičko-logičke (CMP, SHL, SHR, ASHR, ROTL, ROTR, ADC, SBC), memorijske (LOADB, LOADH, STOREB, STOREH), upavljačke naredbe (CALL), upravljačke naredbe (JR), upravljačke naredbe (RET, RETI, RETN, HALT)	6 osnovnih grupa: naredbe load i store (LDR, STR ...), naredbe za obradu podataka (ADD, SUB, MAC, CLZ ...), naredbe grananja (B (skače na petlje) i BL ()poziva potprogram), naredbe za prijenos registara stanja (prijenos iz CPSR ili SPSR u neki registar opće razmjene), koprocesorske naredbe (MRS, MSR...), naredbe za generiranje iznimke (BIC...)
ZAPIS U MEMORIJI	little-endian	little-endian
NAČINI ADRESIRANJA (podcrtano znači u primjeru da se na to odnosi)	registarsko adresiranje (ADD R0, 12, R0; MOVE SR, R3; LOAD R6, (1000)), neposredno adresiranje (ADD R0, 12, R2; MOVE 200, R3; ROTL R2, ZA_TRI, R2), apsolutno adresiranje (LOAD R0, (1200); JP 2000; CALL POTPROG), relativno adresiranje (JR 1200, JR PETLJA), registarsko indirektno adresiranje (JP_CC (R0), CALL (R6)), registarsko indirektno adresiranje s odmakom (LOAD R5, (R0+4); STORE R5, (R6-12); STORE R5, (R6)), implicitno adresiranje (PUSH R5, POP R6)	
KARAKTERISTIKE	5 osnovnih dijelova: ulaz, izlaz, memorija, put podataka i upravljačka jedinica (unutar procesora se nalaze pp i uj); osnovne zadaće procesora: dohvat naredbe, dekodiranje i izvršavanje; interni registri procesora (znači da programer nema izravan pristup njima): podatkovni registar (DR – međusklop između unutrašnjosti procesora i podatkovne sabirnice), adresni registar (AR – međusklop između unutrašnjosti procesora i adresne sabirnice) i instrukcijski registar (IR – registar koji čuva strojni kod naredbe za vrijeme njenog dekodiranja)	Load i store vrste odmaka: Osnovni odmak Reg = mem[BazniReg + Odmak] Predindeksiranje BazniReg = BazniReg + Odmak Reg = mem[BazniReg] Postindeksiranje Reg = mem[BazniReg] BazniReg = BazniReg + Odmak; dodavanjem slova S na kraju naredbi za obradu podataka, mijenja se stanje zastavica (npr MOVS, SUBS ...);

POTPROGRAM	inicijalizira se stog (MOVE 10000, SP), potprogram se poziva naredbom CALL, sprema se vrijednost PC (pokazuje adresu iduće naredbe) na stog, sa PUSH (npr. R0, R1) spremamo registre koje ćemo koristiti, a obrnutim redoslijedom ih vraćamo sa POP (R1, R0) i iz potprograma se izlazi sa naredbom RET	inicijalizira se stog (MOV R13, #0x8100), potprogram se poziva naredbom BL, sprema se vrijednost PC u registar R14 (ili LR), spremamo registre sa STMFD R13!, {R0, R1}, a vraćamo sa LDMFD R14!, {R0, R1} i nakraju izlazimo iz potprograma sa naredbom MOV PC, LR
KRAJ PROGRAMA (naredba za zaustavljanje procesora)	HALT	SWI 123456 ili
VRIJEDNOSTI	brojevi po općenitom su u HEKSADEKADSKOM sustavu	brojevi su po općenitom u HEKSADEKADSKOM sustavu .. u glavnom programu se vrijednosti pišu sa ljestvicom ispred sebe (npr #2)
NEKE STVARI		BRISANJE BITA U STATUS REGISTRU MRS R0, CPSR BIC R0, R0, #0x80 MSR CPSR_C, R0
HAZARDI	Strukturni i upravljački	Strukturni, upravljački i podatkovni

Zastavice općenito:
ništica Z (ako broj nije 0 onda je z=0)
preljev ov (gledaju se najviša 2 bita brojeva koji se prenose i radi se XOR i ako je 1 i 1 onda je to 0)
prijenos (gleda se zadnji broj i ako je 0 onda je prijenos 0)
posudba (ona je suprotno od prijenosa)
predznak (najviši bit rezultata)

GREŠKE:
kod NBC brojeva se javljaju kad je prijenos=1 (a kod oduzimanja se vidi kad je prijenos =0)
kod 2'k brojeva se javljaju kad je preljev=1 (a kod oduzimanja se vidi kad je preljev =1)

LITTLE-ENDIAN (HEXA ZAPISI BROJEVA)

```
MOVE   10000, SP
MOVE   1234, R0
PUSH    R0
```

FFFC	34
FFFD	12
FFFE	00
FFFF	00
10000	

BIG-ENDIAN (HEXA ZAPISI BROJEVA) →od gore prema dolje se čitaju brojevi

```
MOVE   10000, SP
MOVE   1234, R0
PUSH    R0
```

FFFC	00
FFFD	00
FFFE	12
FFFF	34
10000	

Adresiranje u load/store ARM	Osnovni odmak	Predindeksiranje	Postindeksiranje
Neposredni	[R0, #4]	[R0, #4]!	[R0], #4
Registarski	[R7, -R3]	[R7, R3]!	[R7], R3
Registarski s pomakom	[R3, R5, LSL #2]	[R3, R5, LSL #2]!	[R3], R5, LSL #2

POJMOVI:

paritet -> broj jedinica u nekom podatku

'DW -> definira 8 bitni podatak/tke na lokaciji koja je zadana, odnosno ne poravnava ili dijeli

DW, DH, DB -> definiraju 32, 16, 8 bitne podatke i poravnavaju sa adresom djeljivom sa 4, 2, 1

'EQU -> hmm a labeli daje neku vrijednost, da lakše baratamo

'DS -> zauzima, odnosno rezervira mjesta na lokaciji za podatke

64 bitni brojevi, zbrajanje sa carry, slajd 100 03_FRISC i oduzimanje sa posudbom sajd 104

PRIMJERI (što se događa interno):

CALL ADRESA

SP − 4 −> SP (smanji se stog za 4 bajta i vrh stoga sad tamo pokazuje)

PC −> (SP) (vrijednost iduće naredbe se sprema na stog, vrh stoga se ne pomiče normalno)

ADRESA −> PC (upisuje se adresa potprograma u brojilo i izvodi se)

RET

(SP) -> PC

SP + 4 -> SP

Sabirnice prema namjeni dijele se na adresnu, podatkovnu i upravljačku. Sabirnice se mogu dijeliti i na memorijsku, ulazno-izlaznu sabirnicu i sabirnicu specijalne namjene. Prema načinu komunikacije dijele se na asinkrone i sinkrone. Sinkrone koriste CLOCK (takt) za sinkronizaciju, a asinkrone preko zahtjeva odnosno potvrda spremnosti.

PROTOČNA STRUKTURA:

Razina za dohvat	**Razina za izvođenje**
Dohvat nardbe	Izvođenje AL operacije
Dekodiranje naredbe	Spremanje rezultata
Dohvat operanada	

Do trenutka dekodiranja sve naredbe imaju jednaku fazu dohvata:

Rastući brid CLOCK-a: PC → AR

Padajući brid CLOCK-a: PC+4 →PC, (AR)→IR, dekodiranje

uglavnom preko primjera napisati sto je isto, izdvojiti primjere svih i napisati neku tablicu, 5 FISC preedavanj, 23 slajd

TRAJANJE IZVOĐENJA NAREDBI (FRISC):

Jednociklusne – aritmetičko logičke naredbe i registarske naredbe (ADD, CMP, SUBB, MOVE ...)

Dvociklusne – memorijske naredbe i upravljačke naredbe (STORE, LOAD, PUSH, POP, JR, JP, HALT...)

IZNIMKE : ako zadnja naredba traje 1 ciklus, onda dodajemo +1 jedan (zato jer izlazimo iz protočne strukture)

HAZARDI:

Strukturni (u protočnoj strukturi se nemogu tako brzo izvest naredbe, zato dođe do zastoja i da se ukloni treba se čejkati) se pojavljuje kod STORE, LOAD, PUSH i POP (zato su dvociklusne).

Upravljački se pojavljuje kod grananje, odnosno naredbi JP, JR, CALL, RET ...

Podatkovni (FRISC ga nema)

KASNI_5_MINUTA

```
        PUSH R0
        LOAD R0, (KONST) ; 10^9 prolazaka
LOOP    SUB R0, 1, R0    ; 1 takt
        JR_NZ LOOP       ; 2 takta
        POP R0
        RET
KONST   DW    %D  1000000000
```

Ovaj potprogram je napisan za frekvenciju 10 MHz: 5 min = 300 sek = $300*10*10^6=3*10^9$ taktova

Prihvaćanje NEMASKIRAJUĆEG prekida kod FRISCa:
aktivira se priključak IACK (VJ treba deaktivirati INT3), briše se IIF (zabranjivanje daljnih prekida), sprema se PC na stog, deaktivira se IACK i na kraju skok u prekidni potprogram na adresi 0C, tj 0C → PC
Prihvaćanje MASKIRAJUĆEG prekida kod FRISCa:
briše se GIE (zabranjivanje daljnih maskirajućih prekida), sprema se PC na stog i na kraju se s memorijske lokacije na adresi 8 uzima adresa PP i ide se tamo.
PREKIDNI POTPROGRAM kod FRISCa:
sprema se kontekst, otkriva se uzročnik prekida (tj otkriva se koja VJ je izazvala prekid), dojavljuje se VJ da je njen prekid prihvaćen (programski za maskirajuće, preko IACK-a za nemaskirajuće), poslužuje se VJ, obnavlja se kontekst, dojavljuje se da je njezin prekid obrađen i na kraju se vraćamo iz potprograma sa RETI (ako je maskirajući) ili RETN (ako je nemaskirajući)

FRISC	Bezuvjetni prijenos	Uvjetni prijenos	Prekidni prijenos
Lokacije (najčešće u kodu koje stavimo)	BVJ 'EQU 0FFFF1000	UVJ_P 'EQU 0FFFF2000 UVJ_S 'EQU 0FFFF2004 BRISI 'EQU 0FFFF2004	PVJ_POD 'EQU 0FFFF3000 PVJ_IACK 'EQU 0FFFF3004 PVJ_IEND 'EQU 0FFFF3008 PVJ_STOP 'EQU 0FFFF300C
kako se koristi	pomoću petlje,	prvo čekamo da vanjska jedinici	na **nemaskirajućim** prekidima ne

	očitavamo stanje sa te lokacije i ispitujemo da li je na toj lokaciji pritisnuta tipka (ili sta vec treba)	bude spremna (odnosno provjeravamo spremnost), kada je spreman uzmemo ili šaljemo podatak, (ovisno sto treba) i na kraju obrišemo spremnost da se nastavi komunikacija sa vanjskom jedinicom	treba čitati spremnost, jer IACK to sam radi .... na maskirajućim treba prihvaćati prekide (PVJ_IACK) i brisati
Spremnost	ne provjerava	provjerava	

	FRISC-CT	FRISC-PIO	FRISC-DMA
	CTLR 'EQU 0FFFF0000 CTCR 'EQU 0FFFF0004 CTIACK 'EQU 0FFFF0008 CTIEND 'EQU 0FFFF000C *LR gleda samo 16 nižih bitova (sto znaci da treba vise ct-ova ako ocemo do vecih brojeva brojati)*	PIOC 'EQU 0FFFF0000 PIOD 'EQU 0FFFF0004 PIOIACK 'EQU 0FFFF0008 PIOIEND 'EQU 0FFFF000C	
	u CTLR stavimo od kud broji, u CTCR označimo da krene brojati (do nule) (u uvjetnom načinu to služi još za provjeru spremnosti, ali korisnije je to raditi u prekidnom načinu), omogućimo prekid ako je potrebno i onda u prekidnom programu obrišemo spremnost (CTIACK), obavimo sta treba i na kraju javimo da je kraj vj (CTIEND) (VALJDA)	u PIOC upišemo način rada sklopa, omogućimo prekide, zatim u prekidnom programu spremimo kontekst, potvrdimo prihvat prekida (PIOIACK), uzmemo ili izračunamo podatak i šaljemo u sklop (PIOD), kada se obavi što se trebalo, onda se još zabrane daljni prekidi (upisom %B 000 u PIOC) i stavimo nešto bzv u PIOEND na kraju potprograma	

7 načina rada: 1 user i 6 privilegiranih (koristiti ćemo 2, obični i brzi prekidni način); obični koristi R13 i R14 registre, a brzi od R8-R14 .. obični je na adresi 18, a brzi prekid, potprogram je na 1C ... izlazak iz PP se izvodi sa SUBS PC, LR, #4 i time se mijenja PC i CPSR
IZNIMKE: procesor pohranjuje pov. adresu u R14, pa pohranjuje CPSR u SPSR, procesor se prebacuje u željeno privilegirano stanje, procesor započinje s izvođenjem potprograma