Poglavlje 2. Arhitektura procesora

Zajednička karakteristika svih računala je da imaju: procesor, memoriju, ulazno/izlazne uređaje. Ta tri osnovna dijela ima svako računalo.
Procesor je najvažniji jer upravlja računalom i podacima. Sastoji se od **upravljačke jedinice** (control unit) i **puta podataka** (datapath).
Upravljačka jedinca - mozak procesora

- logičke strukture upravljaju radom puta podataka, memorije i ul/iz uređajima prema naredbama programa koji se izvodi

Put podataka - obavlja operacije s podacima

- logičke strukture koje omogućuju privremeno pamćenje podataka i međurezultata, i strukture koje omogućavaju prijenos podataka između dijelova za obradu i pamčenje

Naredba je najmanja cjelina koju procesor može razumjeti, to je niz bitova na temelju kojih on obavlja neke aktivnosti.
Rad procesora se temelji na: Dohvatu naredbe čitanjem iz memorije, Dekodiranju naredbe (prepoznavanje bitova naredbe) i Izvođenju naredbe (obavljanje aktivnosti naredbe).
<u>Razine apstrakcije u opisu arhitekture</u>

- Razina sustava: koriste ju programeri jer ne moraju mnogo znati o načinu kako procesor izvodi procesor već se orijentiraju na algoritam izvođenja
- *Arhitektura skupa naredaba* (ISA): razina apstrakcije između razine programa i razine sklopovlja, uključuje opise svih glavnih karakteristika procesora koji su potrebni da bi se napisali programi u strojnom jeziku koji će se ispravno izvoditi, a sa strane sklopovlja opisuje funkcionalnosti koju sklopovlje treba omogućiti
- *Razina mikroarhitekture*: daje detaljan sklopovski opis arhitekture procesora, načine povezivanja dijelova CPUa te signale potrebne za njihovo upravljanje
- Razina logičkih vrata: potpuni sklopovski opis CPUa koji uključuje podatke o vremenskim kašnjenjima signala unutar sklopa
- Razina rasporeda: fizički opis svih sklopova CPUa koji koristi definiranu tehnologiju izvedbe, prikazuju se svi detalji potrebni za preslikavanje ove razine u fizičko sklopovlje u postupku proizvodnje

<u>Tipovi arh, CPUa s obzirom na pristup memoriji</u>

- Von Neumannova arhitektura: program i podaci smješteni su u jedinstvenu memoriju koja ima samo jednu vezu prema CPUu, CPU ne može dohvatiti naredbu i podatke istovremeno (Von N. usko grlo)
- Harvard arh.: posebna memorija za pohranu programa i posebna za podatke, istovremeno dohvaćanje jedne naredbe i jednog operanda

Memorijski sustav se povezuje CPUom pomoću sabirnica (bus), a to su spojni putovi koji povezuju dijelove računala. Sabirnice mogu biti: ADRESNE(određuju adresu mem. lokacije kojoj se želi pristupiti), PODATKOVNE (prijenos podataka između dijelova računala) i UPRAVLJAČKA (upravlja prijenosom podataka i radom sustava). Memorijski sustav je izgrađen od mem. čipova i adresnog dekodera.
Stog - memorijski sustav koji se zasnivaju na pristupu podacima bez eksplicitno navođenja adrese. Zadnji upisani prvi pročitani (LIFO). Niz tanjura postavljenih jedan na drugi. Pointer uvijek pokazuje na vrh stoga. PUSH (stavlja) i POP (miče).
Tipovi arh. procesora s obzirom na dohvat opranada

- Stogovna arhitektura: Prvo je potrebno učitati operande iz memorije i smjestiti ih u interni stog (stavljaju se na vrh stoga). Zatim se izvodi operacija zbrajanja i rezultat se smješta natrag na vrh stoga. Rezultat se kasnije ponovno čita s vrha stoga i sprema u memoriju. Prednosti: jednostavne i brze naredbe. Nedostatci: međurezultati se teško koriste jer ih je uvijek potrebno prvo zapisati natrag u memoriju a onda opet iz nje dovesti u CPU (velik broj pristupa mem. pri izvođenju operacija)
- Akumulatorska arhitektura: jedan operand je uvijek u posebnom registru koji se naziva akumulator Acc i koji je povezan na arit-log jedinicu, dok se drugi operand čita iz memorije, a rezultat se uvijek sprema natrag u Acc. Prednosti: jednostavnija od stogovne, naredbe su jednostavne u bez puno opcija i jedan operand je u memoriji. Nedostatci: međurezultati (osim zadnjeg) ne mogu se koristiti već sve mora biti pohranjeno u memoriju, pa se često treba pristupati memoriji.
- Arhitektura registar-memorija: nastala iz akumulatorske arh. U CPUu se nalazi skup registara koji su povezani na jedan ulaz ALUa. Jedan operand je uvijek u registru (registri opće namjene) dok se drugi čita iz memorije, a rezultat se zapisuje ponovno u registar.
- Arhitektura registar-registar(load-store): modifikacija predohodne arh. U registar se dovode oba operanda iz memorije te se nakon zbrajanja rezultat sprema natrag u registar i iz njega naredbom store u memoriju. Najzastupljnija arhitektura u današnjim CPUima.

Arhitekture s obzirom na skup naredaba

- CISC: uvođenje procesorskih naredba viših programskih jezika, npr: podijeli operande, spremi rezultat i ostatak dijeljenja, umanji reg za jedan i skoči na početak petlje ako je reg veći od nule, pomnoži matrice u mem, pronađi određeni podatak u bloku mem. Prednosti takvih naredbi su jenostavnije prevođenje programa, ušteda mem zbog manjeg broja naredaba te ubrzanje rada zbog manjeg broja dohvata naredba iz memorije. Karakteristike: velik broj naredaba i njihovih inačica, velik broj načina adresiranja, brojni registri posebne namjene(brojači za petlje, adresiranje, podatke itd.). Projektiranje je zahtjevno i složeno.

- RISC: zasnovana na jednostavnim instrukcijama koje se mogu izvoditi velikom brzinom. Primjeri naredaba: učitaj operand iz mem u reg, zbroji dva podatka iz registra, spremi sadržaj reg u mem, umanji reg za jedan, skoči na neku naredbu. CISC(jedna kompleksna naredba) = RISC(niz jednostavnih naredaba = brže izvođenje). Karakteristike: relativno malen skup jednostavnih naredana, manji broj inačica svake naredbe, mali broj načina adresiranja, velik broj ravnopravnih registara opće namjene unutar CPUa.

Logički **adresni prostor** je raspoloživi opseg adresa koje CPU može generirati na svojim adresnim priključcima. Stvarna fizička veličina memorije koja je izvedena u sustavu a time i fizičkog adresnog prostora u računalu manja je ili jednaka logičkom adresnom prostoru.

Redoslijed zapisa podataka u memoriji: 1234

- Little endian - niži dio podataka na nižu adresu 100 34 Intel
  101 12
- Big endian - viši dio podatka na nižu adresu 100 12 /n 101 34 Motorola

Programsko brojilo nazivamo registar u kojem se nalaze adrese naredbi koje se trebaju izvesti.

Strojni kod naredaba je niz bitova koji jednoznačno određuju naredbu. Nizovi su podijeljeni u polja koja pobliže definiraju način izvođenja naredbe, ili služe za razlikovanje od drugih naredaba, ili za razlikovanje načina adresiranja itd.

Širina strojnog koda naredaba

- Širina naredaba jednaka širini procesorske riječi(sabirnice) (RISC): jedno čitanje iz memorije za dohvat cijele naredbe. Omogućuje jednostavnost i pravilnost arhitekture pri dohvaćanju i dekodiranju naredbe. Nedastatci: širina može ograničavati broj naredaba, način adresiranja, registara itd.
- Nema fiksne širine (CISC): prednost je da nema nikakvih ograničenja na broj i složenost naredaba. Nedostatak čini potraba za brojnim čitanjem iz memorije i komplicirana je arhitektura i postupak dekodiranja.

Naredbe i mikroarhitektura puta podataka procesora FRISC

- Aritmetičko-logičke naredbe: ADD, SUB, AND, OR, XOR
  naredba operand_1 operand_2 operand_3 = “izvedi naredbu između prva dva operanda i stavi rezultat u treći operand”
- Memorijske naredbe: LOAD registar, (adresa); STORE registar, (adresa);
  LOAD čita četri mem lokacije počevši od zadane adrese i stavlja ih u registar
  STORE sprema sadržaj zadanog registra i upisuje ga u četri mem lokacije počevši od zadane adrese; BROJ1 DW 55 - (define word)
- Upravljačke naredbe(grananje i petlje): JP adresa; bezuvjetni skok na naredbu s zadanom adresom (moguće korištenje labela)
  JP_uvjet adresa; ako je uvjet ostvaren naredba se ostvaruje, a ako ne izvodi se sljedeća naredba u programu
- Zastavice i registar stanja: Prijenos - C, Preljev - V (ExILI), Nula - Z (NILI), Predznak - N; Spremaju se u registar stanja (SR registar)

- Uvjeti:

Zapis	Značenje (engl.)		Ispitivani uvjet
ULE	Unsigned Less or Equal	≤	C=0 or Z=1
UGT	Unsigned Greater Than	>	C=1 and Z=0
ULT	Unsigned Less Than	<	C=0
UGE	Unsigned Greater or Equal	≥	C=1
SLE	Signed Less or Equal	≤	(N xor V)=1 or Z=1
SGT	Signed Greater Than	>	(N xor V)=0 and Z=0
SLT	Signed Less Than	<	(N xor V)=1
SGE	Signed Greater or Equal	≥	(N xor V)=0

- Proširenje AL naredaba: drugi operand kao konstanta u ADD (26bit određuje da li se uzimaju registri(0) ili konstanta(1), zadnjih 20bitova za zapis konstante; Logički pomak (SHL/SHR src1, src2, dest), aritmetički pomak (ASHR), rotacija(ROTL/ROTR), src1 je podatak, src2 je broj pomaka, svaki pomaknuti bit se sprema u zastavicu C; ADC i SBC; CMP src1, src2(ili konstanta)(src1-src2);

Unaprijeđenje mikroarhitekture puta podataka FRISCa za izvođenje AL naredaba

Uvođenje registarskih naredaba

- MOVE src, dest

Adresiranje registrom

- Upotrijebiti jedan od općih registara za adresiranje.

Uvođenje odmaka

- LOAD dest, (addreg +- odmak)
- STORE src, (addreg +- odmak)

Prijenos podataka različitih širina

- LOADB čita jedno-bajtni podatak iz mem. lokacije i stavlja ga u najniži oktet registra, gornja tri okteta registra pune se nulama
- STOREB upisuje najniži oktet iz registra u jednu mem. lokaciju
- LOADH čita 16bitni podatak iz mem. lokacije… ^
- STOREH upisuje dva najniža okteta iz registra u dvije uzastopne mem. lokacije
- H - stavlja nulu na najniži bit
- B - ne mijenja adresu
- Običan load/store na dva najniža mjesta dodaje nule

Naredbe za rad sa stogom

- R7/SP pokazivač na vrh stoga
- Stog se puni prema nižim adresama
- PUSH src POP dest; zastavice se ne mijenjaju

Proširenje upravljačkih naredaba

- Adresiranje lokacije skoka registrom: JP(_uvjet) (addreg)
- Relativni skok: JR(_uvjet) adresa; moguć skok za +-0,5MB od trenutne naredbe (pomoću PC)
- Naredbe za potprograme: CALL(_uvjet) adr/adrreg poziv potprograma, adresa povratka se sprema u PC(na vrh stoga); RET povratak iz potprograma RET(I)(N)(_uvjet)
- HALT(_uvjet) prekida rad procesora

Načini adresiranja

- Misli se na načine zadavanja operanada, rezultata ili odredišta skoka u naredbama
- Registarsko adresiranje: podatak ili rezultat nalazi se u jednom od registara CPUa
- Neposredno adresiranje: podatak (broj, a ne adresa) se zadaje neposredno u naredbi kao broj, nakon prevođenja taj podatak je zapisan u strojni kod, moguce koristiti labele
- Apsolutno adresiranje: podatak predstavlja adresu podatka/broja u memoriji; u mem. naredbama adresa se piše u zagradama, a u upravljačkim bez njih, može doći labela
- Relativno adresiranje: koristi se samo u JR, najčešće kao labela (iako ne mora), labela predstavlja adresu odredišta skoka, odmak nije adresa nego odmak od trenutne adrese (dodaje se vrijednosti PCa)
- Registarsko indirektno adresiranje: registar zapisan u zagradama (naglašavaju da se ne radi o običnom reg adresiranju), koristi se u upravljačkim naredbama. U registru se nalazi adresa odredišta skoka.

- Registarsko indirektno adresiranje s odmakom: slično ^ ali se koristi u mem. naredbama, uz registar zadaje se i broj koji predstavlja odmak(+-), adr podatka dobiva se zbrajanjem registra i odmaka, LOAD/STORE
- Implicitno adresiranje: u naredbi se ne piše položaj podatka, već se iz same naredbe zna gdje se on nalazi, PUSH/POP (adresa se nalazi u SP)

3. poglavlje Programiranje procesora FRISC

Asemblerski prevoditelj

- Mnemonički programi > Assembler > Premjestivi strojni kod > Povezivač(biblioteke) > Izvršni program > Punitelj > Apsolutni strojni kod > Memorija

Pravila pisanja programa u ovoj knjizi

- jedna naredba = jedna linija
- POLJE_LABELE POLJE_NAREDBE ; POLJE_KOMENTARA
- POLJE_LABELE obavezno počinje od prvog stupca linije, mora početi slovom, prvih 10 znakova se razlikuje, ona je simboličko ime za adresu
- POLJE_NAREDBE ispred sebe mora imati prazninu bez obzira na sve, umjesto naredbe može stajati i pseudonaredba
- POLJE_KOMENTARA počinje ; i proteže se do kraja linije

Pseudonaredbe su mnemonici koji nemaju veze sa procesorom niti se izvode na njemu, već definiraju kako bi se mogli postaviti zahtjevi kako treba obaviti neke korake tijekom prevođenja u assembleru

- ORG adresa - adresa mora biti broj, a ne labela; zadaje asembleru na koju adresu spremiti program, on se sprema na tu adresu ili prvu sljedeću koja je djeljiva s 4, kod FRISC i ARM naredbe počinju na adresama djeljivim s 4, po defaultu se program stavlja na adresu 0
- LABELA EQU podatak - služi za definiranje vrijednosti konstante s imenom labele, podatak mora biti zadan numerički
- DW podaci - služi za izravan upis riječi (4 oktetea) u memoriju računala, podatak mora biti zadan numerički, ispred može stajati labela, little-endian, prevoditelj uzima podatke i sprema ih u izvršni program od prve sljedeće memorijske lokacije na dalje, tako se zauzima i inicijalizira memorija potrebna za spremanje zadanih podataka
- DH podatak - isto što i DW samo zapisuje pola riječi (2 okteta)
- DB podatak - isto što i DW samo zapisuje bajt
- LABELA DS podatak - zauzima veći broj bajtova u memoriji i incijalizira ih na nulu, ako se labela napiše onda poprima vrijednost adrese prve lokacije u nizu, podatak zadan numerički

- LABELA MACRO parametri ENDMACRO - definira niz naredbi koje se mogu iskoristiti u programu kao jedna velika naredba

Zadana baza je heksadekadska, no broj se može zapisati i u drugim bazama tako da mu se doda prefiks %B ili %D ili %H.
Za razlikovanje brojeva od labela, brojevi će uvijek započinjati znamenkom.

Potprogrami
- pozivaju se naredbom CALL a povratak u glavni program naredbom RET

Prijenos parametara registrima
- programer definira preko kojih registara potprogram prima i vraća rezultate
- pozivatelj potprograma prije poziva mora staviti argumente u zadane registre
- prednosti: brzina prijenosa i jednostavnost
- nedostatci: ograničen broj registara, registri nisu slobodni za korištenje kao parametri, nije moguće rekurzivno pozivanje potprograma zbog korištenja istih registara

Prijenos parametra fiksnim lokacijama
- programer definira fiksne memorijske lokacije s kojih potprogram prima argumente i vraća rezultate
- pozivatelj potprograma prije poziva mora staviti argumente na zadane lokacije
- prednosti: nema ograničenja na broj parametara, registri su slobodni
- nedostatci: sporiji rad, duži i kompliciraniji program, rekurzija nije moguća

Prijenos parametara stogom
- pozivatelj prije poziva mora staviti argumente na stog
- potprogram pretpostavlja da se argumenti nalaze ispod povratne adrese na stogu i stavlja rezultat na stog
- pozivatelj pretpostavlja da se nakon izvođenja rezultat nalazi na vrhu stoga
- pozivatelj uklanja argumente sa stoga

Makronaredbe
- prvo se piše labela pa ključna riječ MACRO i na kraju ENDMACRO
- procesor ih ne prepoznaje već se asemblerski prevoditelj brine za njih
- zauzimaju više memorije, dok su potprogrami sporiji

4. Procesor kao komponenta

Priključci
- svaki čip ima određen broj priključaka čija funkcija ovisi o signalima koji prilaze priključkom (tri vrste sabirnica: pod. adr. i upr.)
- jednostruki (jedan signal) i višestruki

- ulazni, izlazni i dvosmjerni

Tipovi sabirnica

- omogućavaju laku nadogradnju sustava
- memorijske, ulazno izlazne i sabirnice posebne namjene (GPU)
- memorijska povezuje procesor i memoriju, mora biti izrazito kratka kako bi mogla raditi na velikim brzinama
- ulazno izlazne služe za povezivanje UI jedinica s procesorom, veće su duljine i prilagođavaju se brzini rada pojedinih UI jedinica, na njih se može spojiti veći broj takvih sklopova, one se spajaju sa procesorom i memorijom direktno preko zajedničke memorijske i UI sabirnice ili preko posebnog međusklopa
- multipleksirane sabirnice
- sabirnički protokoli: sastoji se od niza sabirničkih transakcija koji su slijed koraka potrebnih da bi se na sabirnici izvela određena operacija (npr. čitanja ili pisanja), transakcije obično sadrže zahtjev koji inicira jedan ili više uređaja i odgovor koji traženi uređaj šalje natrag inicijatoru (unutar zahtjeva može se nalaziti podatak, adresa, naredba i sl.)
- prema načinu komunikacije: sinkrone i asinkrone
- sinkrone su sinkronizirane s clockom, vrlo su kratke, bolje prilagođene za slučaj kad svi uređaji imaju jednaku brzinu, najčešće se koriste za memorijske sabirnice
- asinkrone su složenije za implementaciju: uređaji se sinkroniziraju posebnim protokolom sinkronizacije tzv. rukovanjem (strane koje komuniciraju prelaze na sljedeći korak komunikacije tek kad obje strane potvrde da je prethodni korak dovršen), velika duljina, bolje prilagođene za rad s različitim brzinama uređaja, češće se koriste za UI sabirnice

Priključci i sabirnice procesora FRISC

- V i GND služe za napajanje
- clock: izvor signala obično kristalni oscilator, a signal se vodi i do drugih komponenata
- reset: dovodi procesor u inicijalno stanje, svi registri inicijalizirani na nule, izlazni priključci su u neaktivnom stanju, procesor počinje s dohvaćanjem naredbe s adrese nula
- adr[31:0]: 32bitni skup adresnih priključaka pomoću kojih adresira memoriju i vanjske jedinice
- data[31:0]: 32bitni skup podatkovnih priključaka pomoću kojih se podatak prenosi u/iz procesora
- rd i wr: priključci kojima procesor određuje da li će se na sabirnici izvoditi operacija čitanja ili pisanja, samo jedan može biti aktivan u nekom trneutku
- size[1:0]: priključak određuje veličinu podatka(byte 01, halfword 10, word 11)

- wait: priključak pomoću kojeg sporije memorije ili UI traže od procesora produljenje normalnog ciklusa čitanja ili pisanja podatka
- int[1:0]: priključci preko kojih vanjske jedinice postavljaju zahtjev za prekid
- breq (bus request): priključak preko kojeg DMA-jedinica može zatražiti od procesora da joj prepusti upravljanje nad sabirnicom
- back (bus acknowledge): priključak kojim procesor potvrđuje DMA-jedinici da je prihvatio njen zahtjev za upravljanje sabirnicom

5. Izvođenje naredaba i protočna struktura

Izvođeje naredaba u procesoru

- protočna struktura (pipeline), N puta brže od slijednog izvođenja, T = N+(M-1)

Izvođenje naredaba i protočna struktura procesora FRISC

- razina za dohvat: dohvat naredbe, dekodiranje naredbe, dohvat operanada
- razina za izvođenje: izvođenje operacije, spremanje rezultata (ako je definirano naredbom), upravljačka(neke)
- jednociklusne naredbe: efektivno vrijeme izvođenja im iznosi jedan period clock-a (jedan period), AL naredbe i registarske naredbe
- dvociklusne naredbe: efektivno vrijeme izvođenja im iznosi dva perioda (ciklusa) CLOCK-a, memorijske naredbe

Hazardi

- situacije u kojima procesor ne može izvesti sve faze onih naredaba koje se nalaze u protočnoj strukturi nazivamo hazardi
- strukturni hazard: struktura procesora ne omogućava izvođenje istodobno izvođenje svih tih faza; sve memorijske naredbe(load, store, push, pop) uzrokuju strukturni hazard; naredba na kraju izvođenja mora spremiti podatak natrag u memoriju, a pošto je samo jedna sabirnica, dohvat sljedeće naredbe je na čekanju sve dok se sabirnica ne oslobodi (umeće se bubble)
- upravljački hazard (hazard grananja): naredba koja se nalazi u protočnoj strukturi nije naredba koja se treba treba izvesti; događa se kod grananja kad program nastavlja izvođenje s neke druge adrese umjesto sa sljedeće; naredbe bezuvjetnog skoka (dvociklusne)
- podatkovni hazard: javlja se kad se naredba ne može izvesti jer podaci za njeno izvođenje još nisu spremni (nema ga u FRISCU)

6. Upravljačka jedinica procesora FRISC

Vremenska sinkronoziranje upravljačkih signala

- potrebno je da podatak na ulazu u registar bude stabilan u trenutku kad se aktivira signal clock-a

Dvofazno vremensko vođenje

- nejednolika distribucija signala clk može dovesti do problema pa se uvodi dvofazno vođenje
- susjedni blokovi u protočnoj strukturi upravljani su vremenskim signalom koji je vremenski odmaknut: clk1 i clk2
- kad je neparni upaljen, parni ne radi ništa i osigurava stabilnost podatka, i obrnuto
- nepreklapajuće: signali clk1 i clk2 se uopće ne preklapaju
- preklapajuće: aktivne razine se djelomično preklapaju
- ovakvo vođenje uzrokuje gubljenje performansi pa ga napredni procesori ne koriste

Izvedba upravljačke jedinice procesora FRISC

- upravljačka jedinica izvedena je kao dvostruki stroj s konačnim brojem stanja kako bi procesor osigurao stabilne upravljačke signale za korake koji se izvode na rastući brid signala clk i clk_n; kod kompleksnih procesora je to teško izvesti jer su potrebni tisuće stanja i prijelazi funkcija

Izvedba upravljačkih jedinica mikroprogramiranjem

- upravljačka jedinica je izvedena kao jednostavno računalo unutar računala
- upravljanje radom procesora se obavlja izvođenjem internog mikroprograma koji se sastoji od mikronaredaba koje procesor izvodi interno te one moraju biti jednostavne i izuzetno brze - sve je to dio projektiranja procesora
- mikroprograme nije moguće mijenjati (osim kroz posebne postupke kod nadogradnje zbog sigurnosnih razloga)

7. Računalni sustav

- vanjski uređaji se spajaju na procesor preko posebnih međusklopova koji služe ako posrednici (UI jedinice ili vanjske jedinice)
- vanske jedinice: ulazne/izlazne/dvosmjerne; za prijenos podataka, brojanje impulsa, mjerenje vremena, AD i DA pretvorbu, specijalne namjene itd.

Adresiranje vanjskih jedinica

- memorijsko UI preslikavanje: procesor na jednak način prisupa memoriji i vanjskim jedinicama (LOAD, STORE); FRISC
- izdvojeno UI adresiranje: adresni prostor za memoriju izdvojen je od adresnog prostora za VJ, umjesto LOAD i STORE koristi se IN i OUT

Prijenos podatak između računala i VJ

- programski UI prijenos podataka: svi podaci koji se prenose prolaze kroz procesor; bezuvjetni prijenos: prije prijenosa se ne provjerava je li VJ spremna za pijenos podatka (dioda, zvučnici, relej); uvjetni prijenos: prije prijenosa

svakog podatka provjerava je li VJ spremna za prijenos podatka (printer); uz registar ima i bistabil stanja; prekidni prijenos: VJ procesoru dojavljuje svoju spremnost aktiviranjem posebnog signala koji ima značenje zahtjeva za prekid (interrupt request), a koji može doći u bilo kojem trenutku izvođenja programa; maskirajući i nemaskirajući prekid;
- IIF (internal interrupt flag)

Sklopovski UI prijenos podataka
- prijenos podataka prema VJ se obavlja pod upravljanjem posebnog sklopovlja odnosno vanjske jedinice koja služi za direktan pristup memoriji (DMA)
- podaci koji se prenose ne prolaze kroz procesor; brži od programskog
- načini prijenosa: zaustavljanje procesora: DMA preuzima upravljanje nad sabirnicom i ne vraća upravljanje procesoru sve dok ne prenese sve podatke; krađa ciklusa: DMA preuzme upravljanje nad sabirnicom i prenese samo jedan podatak i odmah vraća upravljanje procesoru; blokovski prijenos: istovjetan je krađi ciklusa uz razliku da kad preuzme upravljanje nad sabirnicom prenese nekoliko podataka (blok)

8. Vanjske jedinice računalnog sustava FRISC

FRISC-CT (Counter Timer)
- služi za brojanje impulsa i mjerenje vremena u računalnom sustavu FRISC
- oslobađa procesor od nepotrebnog čekanja na spore vanjeske događaje ili duge vremenske intervale ili česta posluživanja prekida
- konfigurabilan; ima poseban upravljački registar CR (Control register) koji služi za konfiguraciju CT
- na ulaz CNT se dovodi signal kojeg treba brojati, ili ako želimo brojati neke događaje onda na CNT spajamo signal koji pri pojavi događaja generira pozitivan impuls
- CNT je spojen s internim 16-bitnim brojilom prema dolje DC koji pri pojavi svakog pozitivnog impulsa smanji brojilo za jedan sve dok ne dođe do jedan, a kad dođe do nule:
    1. CT postavlja BS = 1 i generira prekid (ako je omogućen)
    2. Na izlazu ZC (Zero count) CT generira jedan pozitivan impuls
    3. CT automatski kopira vrijedonost iz LR u DC, čime brojilo može ponovo početi brojati impulse
- za konfiguriranje CT pogledati službeni šalabahter ili slajdove

FRISC-GPIO (General Purpose Input/Output)
- omogućuje povezivanje vanjskih uređaja na računalni sustav i čitanje/pisanje podataka na takve uređaje
- konfigurabilan -> CR

- može raditi u četiri načina rada:
    1. ulazni (ulazni, sinkroni)
    2. ispitivanje bitova (ulazni, asinkroni)
    3. izlazni (izlazni, sinkroni)
    4. postavljanje bitova (izlazni, asinkroni)
- kod sinkronog rada se koriste signali READY i STROBE
- za konfiguriranje pogledati šalabahter ili slajdove

FRISC-DMA

- ima mogućnost prijenosa 32-bitnih podataka zaustavljanjem procesora i krađom ciklusa
- izvor i odredište podataka može biti bilo koja kombinacija komponenta
- većina priključaka je dvosmjerna (osim back, breq, int i clock)
- tri registra (adresa izvora, adresa odredišta i brojilo) definiraju početnu adresu s koje će se čitati podaci za prijenos, početnu adresu na koju će se spremati podaci te broj podataka koji se treba prenijeti
- nakon svakog prenesenog podatka brojilo se smanji za jedan i adresa izvora i odredišta se uveća za 4
- nakon što je brojilo došlo do nule, DMA završava prijenos, ako je prijenos bio zadan krađom ciklusa postavlja BS = 1 i postavlja prekid (ako je omogućeno)
- za konfiguriranje pogledati šalabahter ili slajdove